TAMPEREEN TEKNILLINEN YLIOPISTO

SAMULI VAINIO

PILVIPALVELUT JA NIIDEN KÄYTTÖÖNOTTO - ERITYISTARKASTELUSSA AMAZON JA DRUPAL

Diplomityö

Tarkastaja: dosentti Ossi Nykänen
Tarkastaja ja aihe hyväksytty tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 7. marraskuuta 2012

Tiivistelmä

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
VAINIO, SAMULI EERIKKI: Pilvipalvelut ja niiden käyttöönotto - erityistarkastelussa Amazon ja Drupal
Diplomityö, 54 sivua
Helmikuu 2013
Pääaine: Hypermedia
Tarkastaja: dosentti Ossi Nykänen
Avainsanat: Amazon Web Services, AWS, Drupal, CDN, pilvipalvelut

Pilvipalvelut ovat ajankohtainen ja mielenkiintoinen tutkimuksen kohde. Tutkimusten mukaan jo noin puolet suomalaisista isoista organisaatioista käyttävät pilvipalveluita hyödykseen. Pilvipalveluiden käytöllä on mahdollista saavuttaa suuriakin hyötyjä, mutta niiden käyttöön liittyy myös potentiaalisia riskejä. Yksi suurimmista huolenaiheista on palvelun tietoturvallisuus ja yksityisyys. Pilvipalveluiden käyttäjiä huolestuttaa tallettaa yksityistä tai liiketoimintakriittistä tietoa jaettuun palvelinympäristöön. Vastapainona pilvipalvelut tarjoavat kuitenkin erinomaista kustannustehokkuutta, skaalautuvat tarpeen mukaan sekä ovat vikasietoisia.

Tässä diplomityössä tutustutaan pilvipalveluihin, pilvilaskentaan sekä niiden taustalla oleviin teknologioihin. Erityistarkastelussa ovat sisällönjakeluverkot sekä niiden toteutustavat. Työssä toteutetaan kustannustehokkaasti skaalautuva ja vikasietoinen pilvipalvelinympäristö. Palvelu toteutetaan käyttäen Amazon Web Services -palvelualustaa. Luotuun pilvipalvelinympäristöön asennetaan Drupal-sisällönhallintajärjestelmä. Tavoitteena on luoda pilvipalvelinympäristössä toimiva Drupal-sivusto, joka käyttää mahdollisimman hyvin hyväkseen kaikkia pilvipalvelinympäristön tarjoamia resursseja.

Drupal-sivuston käyttöönottoon pilvipalvelinympäristössä liittyy myös haasteita. Järjestelmän virheettömän toiminnan kannalta on oleellista, että kaikki palvelininstanssit toimivat täydellisesti yhdessä. Sillä ei saa olla merkitystä, miltä käytössä olevalta palvelininstanssilta tietoa pyydetään. Jokaisen palvelimen tulee tuottaa sama ulostulo samalle syötteelle. Työssä esitellään tapa, jolla Drupal-sivuston tiedot voidaan tehokkaasti jakaa kaikkien palvelininstanssien kesken ja näin tukea koko järjestelmän rajatonta skaalautumista. Sivuston skaalautuvuutta tuetaan myös sisällönjakeluverkon avulla. Kolmannen osapuolen moduulin avulla saadaan Drupal hyödyntämään käytössä olevaa sisällönjakeluverkkoa.

Työn tuloksissa pohditaan pilvipalvelintoteutuksesta saatavia hyötyjä sekä siihen liittyviä riskejä. Lisäksi vertaillaan pilvipalvelinympäristön ja tavallisen, yhden palvelimen järjestelmän ominaisuuksia keskenään. Vertailussa käytetään hyväksi monikriteeristä päätöksentekomallia.

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
VAINIO, SAMULI EERIKKI: Cloud Services and Implementation - case Amazon and Drupal
Master of Science Thesis, 54 pages
February 2013
Major: Hypermedia
Examiners: adjuct professor Ossi Nykänen
Keywords: Amazon Web Services, AWS, Drupal, CDN, cloud services

Cloud computing is now evolving like never before and is therefore an interesting topic of research. Recent research results reveal that already half of the big Finnish companies have adapted to this new technology. Cloud computing offers many benefits, but it also has some potential disadvantages. One of the biggest concerns about cloud computing are security and privacy. Users might not feel comfortable handing over their private and sensitive information to be stored on the cloud servers. On the other hand, cloud platform offers some huge advantages: they are highly scalable, cost-effective and fault-tolerant.

This thesis takes a closer look at cloud services, cloud computing and the technologies used in implementing them. Also a deeper look is taken into implementation of content delivery networks, as they are a big part of creating a scalable hosting environment. The main goal is to build a scalable and cost-efficient cloud hosting environment. It will be implemented by using Amazon Web Services. Cloud hosting environment will be used to install and host a Drupal web site on it. The goal is to configure Drupal to use efficiently all the resources the cloud platform is offering.

Installing and using Drupal on a cloud environment includes some challenges. As the cloud hosting environment consists of many server instances, it is crucial that all the instances work perfectly together. Every instance should return the same output for any given input. Drupal also requires a third-party module to support content delivery networks. This thesis introduces a method to share the content between the instances, and how to configure Drupal to use the CDN to deliver the content to the end-users.

Results chapter includes a deeper analysis of advantages and disadvantages of using cloud services. It also includes a comparison of a traditional single computer hosting vs cloud environment hosting. The data is analysed with multi-criteria decision analysis method.

Alkusanat

Tämä diplomityö on tehty Inno-W Oy:lle osana selvitystyötä, miten tarjottavien verkkopalveluiden skaalautuvuutta ja vikasietoisuutta voitaisiin parantaa pilvipalveluiden avulla. Diplomityön tulosten pohjalta on helpompi tehdä päätös siitä, missä mittakaavassa pilvipalveluita halutaan operatiivisessa toiminnassa hyödyntää.

Haluan kiittää Inno-W Oy:tä, että sain mahdollisuuden toteuttaa tämän diplomityön osana työtehtäviäni. Erityisesti haluan kiittää Inno-W Oy:n Jouni Renforsia, joka aktiivisesti ohjasi ja tuki diplomityöni kirjoittamista. Suuri kiitos myös työni tarkastajalle ja ohjaajalleni Ossi Nykäselle, jolta sain aina tarvittaessa asiantuntevia ohjeita työn kirjoittamisessa. Kiitos Karoliina Vainiolle työn oikolukemisesta.

Suuri kiitos myös perheelleni ja erityisesti äidilleni sekä kaikille ystävilleni, jotka jaksoivat tukea ja kannustaa diplomityöni kirjoittamisessa.

Tampereella, 5. helmikuuta 2013

_____

SAMULI VAINIO

Sisällys

Kuvat

Taulukot

Lyhenteet ja merkinnät

AJAX (engl. Asynchronous JavaScript And XML) on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.

API (engl. Application programming interface), eli ohjelmointirajapinta on määritelmä, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.

AWS (engl. Amazon Web Services) on Amazonin tarjoama elastinen pilvipalvelinalusta.

CDN (engl. Content Delivery Network), eli sisällönjakeluverkko on kokoelma ympäri internetiä sijoitettuja yhteistyössä toimivia verkkoelementtejä, joiden avulla sisältö replikoidaan useille eri surrogaattipalvelimille ja välitetään näiltä loppukäyttäjälle.

CMS (engl. Content management system), eli sisällönhallintajärjestelmä on yleisnimitys tietojärjestelmälle, joka palvelee koko organisaation sisällönhallintaa sen sijaan, että olisi keskittynyt pelkästään johonkin yksittäiseen osa-alueeseen, kuten verkkopalveluiden hallintaan.

CNAME (engl. Canonical Name record) on nimipalvelimissa käytettävä kenttä, joka on alias johonkin toiseen nimipalvelinkenttään.

CSS (engl. Cascading Style Sheets) on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji.

DDoS (engl. Distributed Denial of Service) tarkoittaa useista lähteistä tapahtuvaa samanaikaista hyökkäystä tiettyä kohdejärjestelmää kohtaan. Hyökkäyksen tavoitteena on lamauttaa verkkopalvelu niin, ettei palvelu ole käytettävissä.

DNS (engl. Domain Name System) on internetin nimipalvelujärjestelmä, joka muuntaa verkkotunnuksia IP-osoitteiksi.

EBS (engl. Elastic Block Store) on Amazonin EC2-instansseissa käytettävä verkkolevypohjainen levyjärjestelmä.

EC2 (engl. Elastic Compute Cloud) on Amazonin elastinen pilvialusta, joka mahdollistaa skaalautuvan pilvilaskentaympäristön luomisen.

HTML (engl. Hypertext Markup Language) on kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä, eli hypertekstiä.

HTTP (engl. Hypertext Transfer Protocol) on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.

IaaS (engl. Infrastructure-as-a-Service) tarkoittaa palvelinten ja palvelinsalien ulkoistamista pilveen.

IOPS (engl. Input/Output Operations Per Second) on Amazonin pilvipalvelinympäristössä käytettävä arvo, joka kertoo käytettävän levyn suorituskyvystä.

JavaScript on pääasiassa Web-ympäristössä käytettävä komentosarjakieli. JavaScriptin tärkein sovellus on mahdollisuus lisätä Web-sivuille dynaamista toiminnallisuutta.

Multi-AZ (engl. Multi Availability Zone) on Amazonin pilvipalvelinympäristössä käytettävä termi, jolla tarkoitetaan palvelun käyttöönottoa useammalla eri Amazonin palvelinkeskuksen saatavuusalueella vikasietoisuuden parantamiseksi.

PaaS (engl. Platform-as-a-Service) tarkoittaa palvelualustan ulkoistamista pilveen, esimerkiksi Google App Engine.

PHP (engl. Hypertext Preprocessor) on erityisesti web-palvelinympäristöissä käytettävä ohjelmointikieli, jolla voidaan luoda dynaamisia web-sivuja.

QoS (engl. Quality of Service) on termi, jolla tarkoitetaan tietoliikenteen luokittelua ja priorisointia.

RDS (engl. Relational Database Service) mahdollistaa relaatiotietokantojen luomisen Amazonin pilvipalvelinympäristössä.

REST (engl. Representational State Transfer) on HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

RRI (engl. Request Routing Infrastructure) on sisällönjakeluverkoissa käytettävä pyynnön ohjaus -infrastruktuuri.

S3 (engl. Simple Storage Service) on Amazonin elastinen levyjärjestelmä, joka mahdollistaa tiedostojen tallettamisen ja noutamisen käyttäen REST- ja SOAP-rajapintoja.

SaaS (engl. Software-as-a-Service) tarkoittaa ohjelmiston hankkimista pilvestä palveluna perinteisen lisenssipohjaisen tavan sijaan.

SOAP (engl. Simple Object Access Protocol) on tietoliikenneprotokolla, jonka pääasiallisena tavoitteena on mahdollistaa proseduurien etäkutsu.

URI (engl. Uniform Resource Identifier) on merkkijono, jolla kerrotaan tietyn tiedon paikka (URL) tai yksikäsitteinen nimi (URN).

URL (engl. Uniform Resource Locator) on URI:n erikoistapaus, jota käytetään osoittamaan WWW-sivuja.

1 Johdanto

Tässä diplomityössä luodaan katsaus *pilvipalveluihin* (engl. cloud services), *pilvilaskentaan* (engl. cloud computing), *sisällönjakeluverkkoihin* (engl. content delivery networks, CDN) sekä siihen, miten Drupal-sisällönhallintajärjestelmä saadaan asennettua pilveen ja käyttämään mahdollisimman hyvin käytettävissä olevia resursseja hyväkseen. Pilvipalveluntarjoajia on useita, mutta tässä työssä keskitytään Amazonin tarjoamaan pilvipalvelukokonaisuuteen, AWS:ään (engl. Amazon Web Services). AWS:n tarjoamat palvelut esitellään kattavasti niiltä osin kuin ne liittyvät työn kontekstiin.

AWS:n avulla luodaan automaattisesti skaalautuva, Linux-palvelimista koostuva pilvipalvelinympäristö. Järjestelmän pohjaksi luodaan yksi Linux-palvelin, johon asennetaan ja konfiguroidaan kaikki tarvittavat ohjelmistot. Tästä palvelimesta otetaan levykuva, jonka avulla automaattinen skaalautuvuus voidaan toteuttaa. Amazonin tarjoamien erilaisten mittareiden avulla palvelinten rasitusta tarkkaillaan, ja tarkasteltavan raja-arvon ylityttyä suoritetaan jokin ennalta määritetty toimenpide. Toimenpiteessä voidaan esimerkiksi käynnistää uusi palvelininstanssi, jolloin pilveen saadaan lisää laskentatehoa. Työssä esitellään lisäksi Drupal-sisällönhallintajärjestelmän asentamista luotuun Amazonin pilvipalvelinympäristöön. Tavoitteena on saada Drupal käyttämään mahdollisimman hyvin hyväkseen kaikkia pilvipalvelinympäristön tarjoamia resursseja. Drupalin ytimestä ei vielä löydy kaikkia pilvipalvelinympäristössä tarvittavia moduuleita, joten työssä esitellään tarkemmin, mitä ytimen ulkopuolisia moduuleita on suositeltavaa käyttää. Työn tavoitteena on luoda mahdollisimman kustannustehokkaasti skaalautuva ja vikasietoinen Drupal-sivusto.

Tutkimuksen kannalta on huomattavaa, että keskittyminen pelkästään Amazonin tarjoamaan pilvipalvelutarjontaan johtaa siihen, että tutkimus ja sen aineisto ovat tapaustutkimusta. Tapaustutkimus on luonteeltaan kertaluonteista ja vaikeasti yleistettävää, koska se keskittyy yhden tietyn ilmiön selittämiseen (Garson, 2008; Järvinen & Järvinen, 2004, ss. 75-82).

1.1 Tutkimuskysymykset

Tässä työssä luodaan laaja katsaus pilvipalveluihin, pilvilaskentaan sekä niiden sovelluksiin. Tärkein tutkimuskysymys onkin siis: “Mitä ovat pilvipalvelut?”. Kysymykseen pyritään vastaamaan hyvin kattavasti, sillä kaikki muut toiminnallisuudet perustuvat tähän. Pilvipalveluiden avulla pyritään tarjoamaan asiakkaalle kustannustehokkaampi, vikasietoisempi ja skaalautuvampi IT-infrastruktuuri. Yksi keskeinen skaalautuvuuteen liittyvä sovellus on sisällönjakeluverkko (engl. Content Delivery Network, CDN). Työssä vastataankin siis kysymykseen “Mitä ovat sisällön-

jakeluverkot?".

Pilvipalveluntarjoajia on useita ja niiden määrä kasvaa koko ajan. Arvioiden mukaan Amazon Web Services (AWS) on tällä hetkellä niistä suurin[1]. Tässä työssä ei pyritä vertailemaan eri pilvipalveluntarjoajia, vaan keskitytään yhteen ja tutustutaan paremmin sen tarjoamiin ominaisuuksiin. Tutkittavaksi pilvipalveluntarjoajaksi valittiin Amazon Web Services, eli työssä vastataan kysymykseen: "Mikä on Amazon Web Services ja mitä se tarjoaa?". Työn toteutusosassa luodaan kustannustehokkaasti skaalautuva ja vikasietoinen pilvipalvelinympäristö käyttäen Amazonin tarjoamia työkaluja. Tässä osiossa pyritään vastaamaan kysymyksiin "Miten Amazonin tarjoamien pilvipalveluiden käyttöönotto tapahtuu?" sekä "Miten pilven laskentakapasiteetti saadaan skaalautumaan automaattisesti käyttötarpeen mukaan?". Lisäksi työssä esitellään esimerkkikokoonpano ja esitetään arvioita kysymykseen "Mitä Amazonin pilvipalvelinympäristön käyttö maksaa?".

Työssä toteutetun skaalautuvan pilvipalvelinalustan päälle asennetaan ja konfiguroidaan Drupal-sisällönhallintajärjestelmä. Yhden palvelimen paikallisen asennuksen ja pilvipalvelinympäristössä usean eri palvelininstanssin avulla toimivien järjestelmien välillä on suuria eroja. Pilvipalvelinympäristössä on otettava huomioon eri instanssien välillä jaettavat tiedostot, jotta sivusto toimii yhdenmukaisesti. Lisäksi skaalautuvuuteen halutaan kiinnittää erityistä huomiota jakamalla Drupalin staattiset tiedostot loppukäyttäjällä käyttäen sisällönjakeluverkkoa. Näihin liittyy tutkimuskysymys "Miten Drupal saadaan hyödyntämään kaikkia käytettävissä olevia pilvipalvelinresursseja mahdollisimman tehokkaasti?".

1.2 Työn tavoitteet ja rajaukset

Työn tavoitteena on luoda syvällinen katsaus pilvipalveluihin sekä niiden taustalla vaikuttaviin teknologioihin. Erityistarkastelussa ovat sisällönjakeluverkot, joiden teknistä toteutusta esitellään seikkaperäisesti. Pilvipalveluiden käyttäjien määrä on viime vuosina kasvanut huomattavasti. Pilvipalveluiden avulla on mahdollista saavuttaa huomattavaakin kustannussäästöä sekä muita skaalautuvuuteen liittyviä hyötyjä, mutta siihen liittyy myös potentiaalisia vaaroja. Tässä työssä tavoitteena onkin selvittää tarkemmin, mitä hyötyjä ja haittoja pilvipalveluiden käyttöön liittyy. Vaikka pilvipalveluiden avulla saavutettaisiin huomattaviakin etuja, niin on kuitenkin mahdollista, ettei niitä pystytä kuitenkaan erilaisten rajoitusten vuoksi hyödyntämään. Tällaiset rajoitukset liittyvät usein salaisen tai esimerkiksi bisneskriittisen tiedon tallettamiseen. Lisäksi työssä esitellään tarkemmin Amazon Web Services -palveluntarjoajaa ja sen palveluvalikoimaa. Tavoitteena on tarjota lukijalle kaikki oman skaalautuvan ja vikasietoisen pilvipalvelinjärjestelmän luomisessa

[1] `http://gigaom.com/2012/03/14/amazon-is-no-1-whos-next-in-cloud-computing/`

tarvittava tieto. Luotuun Amazonin pilvipalvelinympäristöön asennetaan Drupal-sisällönhallintajärjestelmä ja asetetaan se käyttämään mahdollisimman hyvin hyödykseen kaikkia saatavilla olevia pilvipalvelinresursseja. Työn tavoitteena on siis luoda skaalautuva ja vikasietoinen, Amazonin pilvipalvelinympäristössä toimiva Drupal-sivusto.

Tässä työssä esimerkkitoteutus ja toteutustekniikat rajataan Amazonin tarjoamiin pilvipalveluihin. Amazonin pilvipalveluiden esittely rajataan keskeisesti Drupal-sisällönhallintajärjestelmään liittyviin tekniikoihin, eli muun muassa Linux-pohjaisiin www- ja tietokantapalvelimiin, näihin liittyviin kuormantasausratkaisuihin sekä jaetuille resursseille. Useat toteutettavat ominaisuudet eivät kuitenkaan ole alustariippuvaisia, eli ne toimivat sovellettuina myös esimerkiksi Windows-pohjaisessa järjestelmässä.

1.3 Työn rakenne

Työn alussa esitellään pilvipalveluiden teoriaa sekä niihin liittyviä teknologioita. Luku 2 keskittyy siis puhtaasti pilvipalveluiden teoriaan. Luvussa 3 luodaan katsaus sisällönjakeluverkkoihin. Sisällönjakeluverkkojen yleisen teorian lisäksi luvussa luodaan syvällinen katsaus niiden tekniseen toteutukseen. Luvussa 4 esitellään Amazon Web Services -palveluntarjoajan tarjoamia palveluita.

Työssä toteutettu skaalautuvan ja vikasietoisen pilvipalvelinalustan luominen ja käyttöönotto esitellään luvussa 5. Luvussa 6 esitellään Drupal-sisällönhallintajärjestelmän käyttöönotto aikaisemmin luodussa pilvipalvelinympäristössä.

Lopuksi luvussa 7 vertaillaan tavallisen virtuaalipalvelimen ja pilvipalvelintoteutuksen eroja ja yhtäläisyyksiä. Lisäksi pohditaan Drupalin soveltuvuutta pilvipalvelinympäristöön.

2 Pilvipalvelut ja pilven toteutusmallit

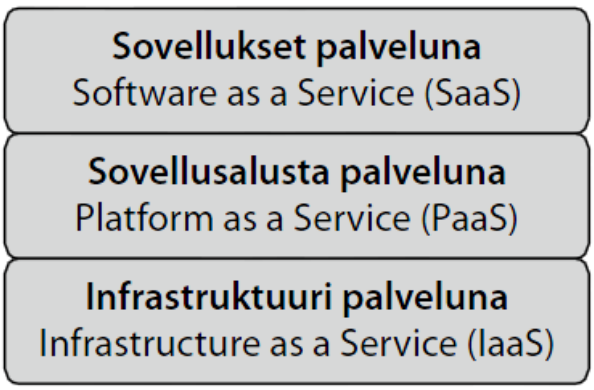
Tässä luvussa määritellään pilvipalvelu, pilvilaskenta sekä esitellään pilvipalveluiden kolme eri tasoa. Niiden lisäksi esitellään erilaiset pilvipalveluiden toteutusmallit, virtualisointitekniikat sekä pohditaan mahdollisia pilvipalveluiden haasteita.

2.1 Pilvi ja pilvilaskenta

Yhtä yleisesti hyväksyttävää määritelmää käsitteelle *pilvilaskenta* (engl. cloud computing) ei ole. Käsitettä *pilvi* (engl. cloud) käytetään kielikuvana, jolla viitataan internetiin ja pilvipalveluilla puolestaan tarkoitetaan mallia, missä tietotekniikkaresursseja (tietoliikenneyhteydet, laskenta- ja tallennuskapasiteetti, sovellukset sekä palvelut) tarjotaan verkon välityksellä käyttöön ilman, että käyttäjän tarvitsee tietää, missä resurssit sijaitsevat, tai huolehtia niiden toiminnasta ja ylläpidosta. (Salo, 2010, s. 16). Buyya et al. (2009, s. 601) ovat määritelleet pilven artikkelissaan seuraavasti: "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." Tämän määritelmän mukaan pilven toiminnassa oleellisia asioita ovat toisiinsa kytketyt, virtualisoidut tietokoneet, jotka tarjotaan dynaamisesti asiakkaalle. Kirjallisuudessa ja artikkeleissa esitettyjä määritelmiä on useita, mutta niissä toistuvat usein samat ominaisuudet. Useimmissa määritelmissä esiintyvät pilven skaalautuvuus, verkkokeskeisyys sekä käyttöajan mukaan tapahtuva laskutus.

2.2 Pilvipalvelumallit

Tarkasteltaessa pilvipalveluita palveluarkkitehtuurina on pilvipalvelumalleja tunnistettavissa useita. Palveluarkkitehtuuri jaetaan yleisesti kolmeen kerrokseen:



Kuva 2.1: Pilvipalveluarkkitehtuurin kolme kerrosta. (Salo, 2010, s. 22)

Infrastruktuuri (IaaS) luo pohjan *palvelualustalle* (PaaS), jonka päälle voidaan rakentaa *sovelluksia* (SaaS). Jako kolmeen ei kuitenkaan ole ainut tapa tarkastella palveluarkkitehtuuria. Omiksi kokonaisuuksikseen voidaan irrottaa esimerkiksi tallennustila palveluna (Storage-as-a-Service), tietoturvapalvelut palveluna (Security-as-a-Service) tai viestintä palveluna (Communication-as-a-Service). Yhteisenä akronyyminä kaikelle palvelullistamiselle ylipäänsä voidaan käyttää lyhennettä XaaS eli X-as-a-Service. (Salo, 2010, s. 22). Seuraavaksi esitellään tarkemmin kaikki kolme yleisintä, kuvassa 2.1 esitettyä pilvipalveluarkkitehtuurikerrosta.

2.2.1 Sovellus palveluna (SaaS)

Sovellukset palveluna (engl. Software-as-a-Service) on käsitteistä tutuin ja ollut käytössä jo vuosia (Salo, 2010, s. 22). Sovellus toimii rajapintana pilven ja erilaisten päätelaitteiden välillä. Se on loppukäyttäjälle näkyvin kerros ja sen kautta laskenta voidaan siirtää suoritettavaksi pilveen. (Youseff et al., 2008, s. 22).

Tämän arkkitehtuurimallin avulla yksi tuotantoympäristö voidaan jakaa useamman asiakkaan kesken ja palvelimella voidaan ajaa samanaikaisesti useita ohjelmainstansseja. Jaetun tuotantoympäristön avulla saavutetaan kustannustehokkuuden lisäksi myös usein huomattavaa parannusta palvelun laadussa ja saavutettavuudessa, kun käytössä on luotettavampi ja nopeampi verkkoyhteys. Vanhan ohjelmistomallin mukaan ensin ostettiin ohjelmistolisenssi, maksettiin ylläpitosopimuksesta sekä jouduttiin läpikäymään useita aikaa vieviä ja kalliita ohjelmistopäivityksiä, jotta palvelun käyttäminen onnistui. Tämä turhautti monia asiakkaita ja nämä asiakkaat ovatkin olleet innokkaita vaihtamaan uuteen, käyttöperusteisen laskutuksen järjestelmään. Monet asiakkaat uskovatkin, että heillä on uuden ohjelmistomallin mukaan enemmän vaikutusvaltaa asiakassuhteessa, kun he voivat helposti vaihtaa palveluntarjoajaa, mikäli palveluntarjoajan tarjoama palvelun laatu ei tyydytä. (Dubey & Wagle, 2007, ss. 2-3).

2.2.2 Sovellusalusta palveluna (PaaS)

Sovellusalusta palveluna (engl. Platform-as-a-Service) tarjoaa sovelluskehittäjälle ohjelmointiympäristön, jonka avulla sovelluskehittäjä pystyy luomaan oman ohjelmistonsa ja tarjoamaan tätä asiakkaalle palveluntarjoajan palvelinympäristöstä. Palveluntarjoaja tyypillisesti kehittää työkalut ja luo standardit ohjelmistokehitykselle omassa palvelinympäristössään. (Mather et al., 2009, s. 19). Sovellusalustojen avulla saavutettavia etuja ovat automaattinen skaalautuvuus, kuormantasaus sekä integroituminen muihin palveluihin (Youseff et al., 2008, s. 4).

Esimerkki yhdestä tämän hetken suurimmista sovellusalustoista on Google

App Engine[2]. Google (2012) listaa App Enginensä hyödyiksi muun muassa automaattisen skaalautuvuuden, luotettavuuden, korkean suorituskyvyn sekä Googlen infrastruktuurin turvallisuuden. Sovellusalustoilla on kuitenkin myös paljon rajoituksia. Google App Engine esimerkiksi tukee vain Java- ja Python-ohjelmointikieliä. Organisaation käytössä oleva ohjelmisto on saatettu ohjelmoida jollakin eri ohjelmointikielellä, jolloin sen siirtäminen Google App Engineen vaatii ohjelmiston uudelleenohjelmoinnin käyttäen tuettuja ohjelmointikieliä. Tämä aiheuttaa palvelun käyttöönotossa niin ajallista viivettä kuin myös kuluja.

2.2.3 Infrastruktuuri palveluna (IaaS)

Infrastruktuuri palveluna (engl. Infrastructure-as-a-Service) tarkoittaa laitteiston ja siihen liittyvän ohjelmiston ja perustoiminnallisuuksien tarjoamisen pilven ylläpitämisen mahdollistamiseksi (IaaS Definition, 2012). Palveluntarjoajan tehtävänä on siis ainoastaan pitää palvelinkeskus toimintakunnossa ja asiakas vastaa itse ohjelmistoista, skaalautuvuuden hallinnasta ja muista käytössä olevien resurssien riittävyydestä.

Sovellus palveluna (SaaS) sekä sovellusalusta palveluna (PaaS) rajoittivat paljon sitä, miten palvelua voidaan käyttää, mutta siirsivät myös paljon ylläpitovastuuta asiakkaalta palveluntarjoajalle. Infrastruktuuri palveluna (IaaS) -mallin myötä ylläpitovastuuta siirretään sen sijaan asiakkaalle, mutta samalla myös palvelun rajoitteet pienenevät. Asiakas voi IaaS-mallissa valita ohjelmistojen suhteen lähes kaiken, alkaen käyttöjärjestelmän valinnasta. Käyttöjärjestelmään on mahdollista asentaa siis mikä tahansa asiakkaan toivoma tietokantaohjelmisto tai käyttää mitä tahansa asiakkaan haluamaa ohjelmointikieltä ohjelmiston toteutuksessa.

Amazon EC2[3] on IaaS-mallin mukainen järjestelmä. Amazon EC2 tarjoaa käyttöjärjestelmävaihtoehdoiksi joko Microsoft Windowsia tai Linuxia. Molemmista käyttöjärjestelmistä on tarjolla erilaisia, joko Amazonin tai yhteisön tarjoamia versioita, jotka on suunniteltu erilaisiin käyttötarpeisiin. Esimerkiksi BitNamin[4] kautta on saatavilla valmiita levykuvia, jotka helpottavat Drupalin käyttöönottoa Amazonin pilvipalvelinympäristössä. Näiden levykuvien myötä järjestelmästä löytyvät esiasennettuna kaikki Drupalin käytössä tarvittavat ohjelmistot.

2.3 Pilven toteutusmallit

Pilven toteutusmallit voidaan jakaa karkeasti kahteen pääluokkaan: *yksityiseen pilveen* (engl. private cloud) ja *julkiseen pilveen* (engl. public cloud). Julkisen pilven

[2] `https://developers.google.com/appengine/`
[3] `http://aws.amazon.com/ec2/`
[4] `http://bitnami.org/stack/drupal`

tapauksessa pilvipalvelut tarkoittavat tietokonekapasiteetin ja -palveluiden ostamista ja hyödyntämistä joustavasti palveluna internetin välityksellä. Yksityinen pilvi on tarkoitettu organisaation sisäiseen käyttöön. Tällöin palvelut voidaan toimittaa esimerkiksi yrityksen oman intranetin kautta palomuurin sisäpuolella tai se voi myös olla ulkoistettu palveluntarjoajan konesaliin. (Vrt. IBM, 2012; Grossman, 2009, s. 24). Pilven toteutusmallit voidaan kuitenkin nykyisin jakaa jo neljään luokkaan. Aikaisemmin esiteltyjen yksityisen ja julkisen pilven lisäksi myös *yhteisöpilvi* (engl. community cloud) sekä *hybridipilvi* (engl. hybrid cloud) voidaan eriyttää omiksi toteutusmalleikseen. (Dillon et al., 2010, s. 28). Kuvassa 2.2 on esitetty yksityisen pilven, julkisen pilven ja yhteisöpilven rakenne. Hybridipilvi muodostuu näiden kolmen yhdistelmästä. Kaikki neljä toteutusmallia esitellään tarkemmin myöhemmin tässä luvussa.

Kuva 2.2: Pilven toteutusmalleja (How CRM Works, 2010)

Suurin osa pilvilaskennan infrastruktuurista koostuu luotettavista palveluista, jotka toimitetaan asiakkaalle datakeskusten ja tarkoitusta varten pystytettyjen palvelinten avulla hyödyntäen useita erilaisia virtualisoinnin tasoja. Palveluiden käyttäminen on mahdollista mistä tahansa, kunhan verkkoyhteys infrastruktuuriin on käytettävissä. Asiakkaalle pilvi näkyy yhtenä yhteyspisteenä. Avoimet standardit ovat kriittinen vaatimus pilvilaskennan kehittymisen ja kasvamisen kannalta. Avoimen lähdekoodin ohjelmistot ovat tarjonneet myös perustan monille pilvilaskennan toteutuksille. Esimerkiksi AWS (engl. Amazon Web Services) hyödyntää Xen-teknologiaa. (Mather et al., 2009, s. 23). Taulukossa 2.1 on esitetty pilvilaskennalla saavutettavia hyötyjä tavanomaiseen ratkaisuun nähden.

Taulukko 2.1: Pilvilaskenta asiakkaan näkökulmasta (Mather et al., 2009, s. 26)

Dedikoitu / tavanomainen tietojärjestelmä	**Pilvilaskenta**
Suuret etukäteisinvestoinnit uusille järjestelmille	Pienet etukäteisinvestoinnit uusille järjestelmille; laskutus palvelun käyttömäärän mukaan
Luotettavan infrastruktuurin suunnittelu ja toteuttaminen kallista	Luotettavuus on sisäänrakennettuna pilvipalveluarkkitehtuurissa
Tietojärjestelmän ympäristö erittäin monimutkainen	Modulaarinen ympäristö
Monimutkainen arkkitehtuuri	Ei infrastruktuuria

2.3.1 Yksityinen pilvi

Yksityiset pilvet ja *sisäiset pilvet* ovat termejä, joita käytetään kuvaamaan pilvilaskentaa sisäisessä verkossa. Sisäisen verkon avulla saavutetaan pilvilaskennan hyötyjä, mutta voidaan poissulkea julkisen pilven sudenkuoppia. Sisäisen verkon avulla voidaan poistaa huolia, jotka liittyvät tietoturvallisuuteen, yrityksen hallintoon sekä luotettavuuteen. Organisaatioiden pitää kuitenkin ostaa ja rakentaa oma sisäinen pilvensä itse, jolloin palvelun aloitus- ja ylläpitokustannukset ovat huomattavasti suuremmat. Yksityisen pilven omistajaorganisaatio on itse vastuussa oman pilvensä toiminnasta. (Mather et al., 2009, s. 23). Yksityiset pilvet voidaan vielä lajitella omiksi kokonaisuuksikseen pilveen liittyvien toimijoiden perusteella. Taulukossa 2.2 on esitelty esimerkki tällaisesta jaottelusta.

Taulukko 2.2: Yksityisen pilven eri tyypit (Mather et al., 2009, s. 24)

Yksityisen pilven tyyppi	**Selitys**
Dedikoitu	Yksityinen pilvi, jonka konesalipalvelut tarjotaan yrityksen omistamasta toisesta datakeskuksesta ja palvelun operoinnista vastaa paikallinen IT-osasto
Yhteisö	Yksityinen pilvi, jonka konesalipalvelut sijaitsevat kolmannen osapuolen tiloissa. Konesalipalvelut omistaa, hallinnoi ja operoi kolmas osapuoli, joka on sitounut omistajaorganisaation kanssa yksilöityyn palvelutasosopimukseen (engl. Service Level Agreement), joka kattaa muun muassa tietoturvallisuudesta huolehtimisen sovitulla tavalla
Hallittu	Yksityisen pilven omistaa organisaatio itse, mutta sen hallinnasta vastaa kolmas osapuoli

2.3.2 Julkinen pilvi

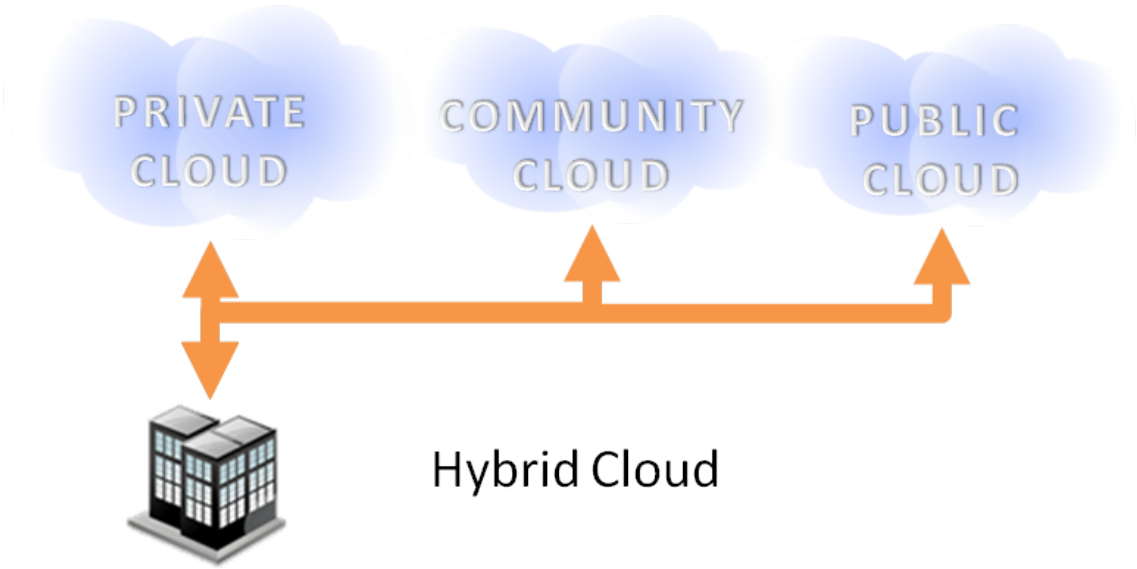
Julkiset pilvet (tai *ulkoiset pilvet*) kuvaavat pilvilaskentaa yleisimmässä ja kansanomaisimmassa muodossaan, jossa resursseja tarjotaan dynaamisesti ja itsepalvelumuotoisesti internetin kautta. Resursseja tarjotaan web-ohjelmiston tai -sovelluksen muodossa joltakin ulkoiselta, kolmannen osapuolen palvelimelta. Laskutus tapahtuu käytön mukaan ja kokonaislasku muodostuukin useista erilaisista kustannuslähteistä. Julkista pilveä hallitsee ja operoi jokin kolmas osapuoli yhdestä tai useammasta datakeskuksesta. Pilvipalveluita tarjotaan useille asiakkaille käyttäen yhtä yhteistä infrastruktuuria. Julkisissa pilvissä vastuu infrastruktuurin tietoturvallisuudesta ja päivittäisestä operoinnista on siirretty palveluntarjoajalle. (Mather et al., 2009, s. 23).

2.3.3 Yhteisöpilvi

Yhteisöpilvessä useat organisaatiot yhdessä muodostavat ja jakavat saman pilvi-infrastruktuurin. Yhteisen infrastruktuurin myötä myös pilven toimintamallit, vaatimukset, arvot ja muut pilveen liittyvät seikat ovat yhteisiä. Yhteisöpilvessä päätöksenteko tapahtuu demokraattisessa tasapainossa ja sen avulla yhteisöt saavuttavat kustannustehokkaasti skaalautuvan pilven. Pilvi-infrastruktuurin ylläpito voi olla ulkoistettu kolmannelle osapuolelle tai jokin yhteisössä mukana oleva taho voi hallinnoida sitä itse. (Dillon et al., 2010, s. 28).

2.3.4 Hybridipilvi

Hybridipilvi on kahden tai useamman julkisen, yksityisen tai yhteisöpilven sekoitus. Hybridipilven avulla organisaatio voi esimerkiksi suorittaa ydinliiketoimintansa kannalta tärkeät sovellukset sisäisessä pilvessä ja tietoturvallisuuden kannalta pienemmän prioriteetin operaatiot julkisessa pilvessä. (Mather et al., 2009, s. 25). Esimerkiksi yrityksen henkilöstöhallinnon ohjelmisto voisi käyttää sisäistä pilveä, jottei henkilö- tai palkkatietoja pääse vahingossa vuotamaan organisaation ulkopuolelle. Kuvassa 2.3 on esitetty esimerkki hybridipilven rakenteesta.



Kuva 2.3: Hybridipilvi (How CRM Works, 2010)

2.4 Virtualisointi

Virtualisointi on yksi keskeisimmistä pilvipalvelun taustateknologioista. Scheffy (2007, s. 601) jakaa virtualisointitekniikat kolmeen pääluokkaan:

- Täysvirtualisointi,
- Paravirtualisointi sekä
- Laitteistoavusteinen virtualisointi.

Kaikkia kolmea lähestymistapaa yhdistää se, että ne tarvitsevat *hypervisorin* toimiakseen. Hypervisor tunnetaan myös nimellä virtuaalikoneohjain (engl. virtual machine manager, VMM). (Goldberg, 1973, s. 16).

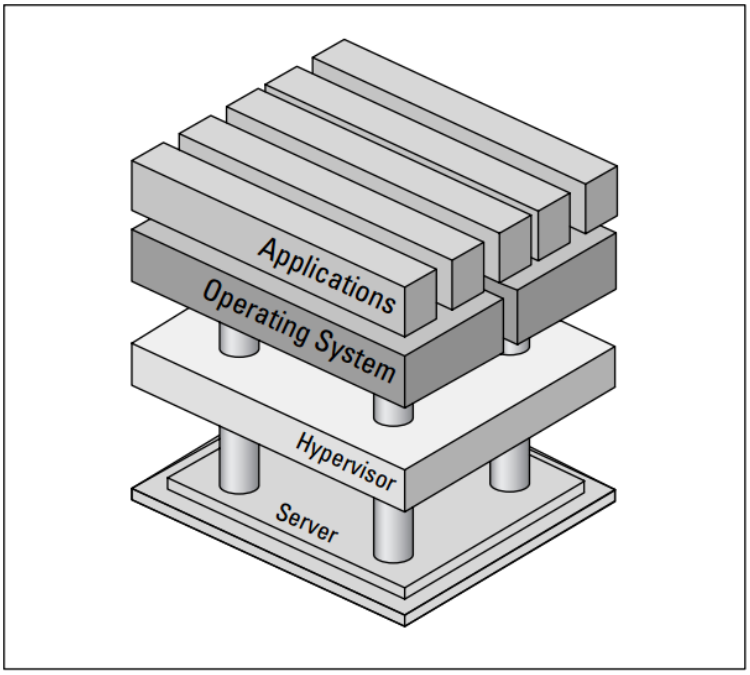
Douglas & Gehrmann (2009, s. 10) toteavat, että virtualisointi on järjestelmän abstrahointia, jossa virtualisointikerros käsittelee ja oman rajapintansa avulla tarjoilee virtuaaliset resurssit asiakaskerrokselle. Virtualisoinnin avulla erotetaan laitteisto käyttöjärjestelmästä, jolloin asiakkaat voivat asentaa virtualisoidulle palvelimelle haluamansa käyttöjärjestelmän. Tämä mahdollisuus on luonnollisesti tarjolla vain infrastruktuuri palveluna -arkkitehtuurissa (IaaS), jossa asiakas vastaa itse käyttöjärjestelmästä.

2.4.1 Täysvirtualisointi

Täysvirtualisoinnilla tarkoitetaan tilannetta, jossa emuloidaan suoraan laitteistoa eikä käyttöjärjestelmää. Tällöin hypervisor ottaa kiinni jokaisen I/O-kyselyn ja käsittelee ne itse. Hypervisor vastaa käyttöjärjestelmälle samoin kuin itse laitteisto

olisi vastannut, eli käyttöjärjestelmä ei tiedä tuliko vastaus hypervisorilta vai itse laitteistolta. Scheffy (2007, s. 22). Kuvassa 2.4 näkyy hypervisorin rooli virtualisoidun käyttöjärjestelmän ja palvelimen välillä.

Tämän virtualisointitekniikan etu on se, että virtualisoidun käyttöjärjestelmän näkökulmasta hypervisor on näkymätön. Käyttöjärjestelmä ei tiedä, kommunikoiko se suoraan laitteiston vai hypervisorin kanssa. Tälle ei luonnollisesti ole käyttöjärjestelmän puolesta minkäänlaista merkitystä, mutta sen avulla käyttöjärjestelmät toimivat suoraan ilman muutoksia tai erikoisohjelmia. Täysvirtualisoinnin haittapuolena sen sijaan on sen aiheuttama ylimääräinen rasitus järjestelmälle. Erityisesti paljon I/O-kyselyitä tuottavassa ympäristössä saattaa järjestelmän suorituskyky laskea huomattavastikin. Scheffy (2007, s. 21-22).



Kuva 2.4: Hypervisorin rooli täysvirtualisoinnissa (Scheffy, 2007, s. 23)

2.4.2 Paravirtualisointi

Scheffyn (2007, ss. 23-24) mukaan *paravirtualisoinnissa* tai *osittaisvirtualisoinnissa* (engl. partial virtualization) järjestelmään tulee asentaa erillinen isäntäkäyttöjärjestelmä, jonka avulla tarjotaan rajapinta virtuaalikoneille. Paravirtualisoinnin avulla eliminoidaan täysvirtualisoinnissa syntyvät ylimääräiset I/O-kyselyt (engl. overhead). Tällöin siis virtualisoitu käyttöjärjestelmä tietää, että se kommunikoi isäntäkäyttöjärjestelmän kanssa hypervisorin kautta ja antaa näin hypervisorin huijata itseään. Paravirtualisoinnin etuna on se, että se saattaa olla nopein virtualisointitekniikka. Haittapuolena on kuitenkin tarve käyttöjärjestelmän erikoisversiolle, jossa paravirtualisoinnin tarpeet on huomioitu.

Täysvirtualisoinnissa hypervisor sijoittui palvelimen ja virtualisoidun käyttöjärjestelmän väliin. Paravirtualisoinnissa rakenne on muuten samanlainen kuin kuvassa 2.4, mutta hypervisor on kiinni käyttöjärjestelmässä. Sovellukset, käyttöjärjestelmä ja hypervisor muodostavat siis yhteisen kokonaisuuden.

2.4.3 Laitteistoavusteinen virtualisointi

Laitteistoavusteinen virtualisointi toimii kuin täysvirtualisointi, eli se tarjoaa suoran yhteensopivuuden virtualisoidun käyttöjärjestelmän kanssa. Virtuaalikoneen käyttöjärjestelmä ei siis tarvitse mitään erillisiä ohjelmistoja. Palvelimen prosessorin sen sijaan pitää tukea virtualisointia. Tämän menetelmän avulla virtualisoinnista syntyvä ylimääräinen rasitus on saatu minimoitua, eikä enää ole havaittavissa kuvan 2.4 tapaista erottelua käyttöjärjestelmään, hypervisoriin ja palvelimeen. Sen sijaan järjestelmä voidaan nähdä yhtenäisenä kokonaisuutena. Barrett & Kipper (2010, s. 14).

2.5 Yhteenveto

Pilvipalveluille ja pilvilaskennalle löytyy kirjallisuudesta useita erilaisia määritelmiä. Pilvilaskennalle ei kuitenkaan ole olemassa yhtä yleisesti hyväksyttävää määritelmää. Erilaisista kirjallisuudesta löytyvistä määritelmistä löytyvät kuitenkin usein termit skaalautuvuus, virtualisointi sekä käyttöperustainen laskutus.

Pilvipalvelut jaetaan kolmeen pääryhmään: sovellus palveluna (SaaS), sovellusalusta palveluna (PaaS) sekä infrastruktuuri palveluna (IaaS). Sovellus palveluna -mallissa asiakas ostaa käyttöoikeuden valmiiseen sovellukseen. Sovellusalusta palveluna -mallissa palveluntarjoaja tarjoaa asiakkaalle ympäristön, jossa asiakkaalla on mahdollisuus tietyin rajoittein toteuttaa oma sovellus. Infrastruktuuri palveluna -mallissa puolestaan palveluntarjoaja tarjoaa laitteiston ja asiakkaan vastuulle jäävät koko käyttöjärjestelmä ja sen sovellukset.

Pilven toteutusmallit voidaan jakaa kahteen pääluokkaan: yksityiseen ja julkiseen pilveen. Lisäksi on olemassa näiden variaatioita, kuten yhteisöpilvi ja hybridipilvi. Julkisessa pilvessä resursseja tarjotaan dynaamisesti ja itsepalvelumuotoisesti internetin kautta. Yksityiset pilvet on puolestaan tarkoitettu organisaatioiden sisäiseen käyttöön. Yhteisöpilvessä useat organisaatiot voivat jakaa yhteisen pilven, jolloin saavutetaan parempi kustannustehokkuus. Hybridipilvessä puolestaan osa palveluista tarjotaan julkisesti ja osa taas on vain organisaation sisäiseen käyttöön, esimerkiksi palkanlaskentaan.

Tärkein pilvipalveluiden taustalla oleva teknologia on virtualisointi. Järjestelmätason virtualisoinnin avulla yhdestä fyysisestä palvelimesta voidaan luoda useita virtuaalisia palvelininstansseja. Virtualisointi mahdollistaa myös sen, että

virtuaalikoneeseen on mahdollista asentaa eri käyttöjärjestelmä kuin mikä isäntäkoneessa on. Virtuaalipalvelimelle voidaan allokoida vain sen verran resursseja kuin se tarvitsee, jolloin yhdellä tehokkaalla fyysisellä palvelimella voidaan ajaa kymmeniä virtuaalipalvelimia.

3 Sisällönjakeluverkot

Tässä luvussa tutustutaan sisällönjakeluverkkoihin ja niiden toimintaan. Sisällönjakeluverkkojen toimintaperiaatteet esitetään kattavasti niin teoreettisesti kuin teknisestikin.

3.1 Yleiskatsaus

Internetin nopeaan kasvuun on vaikuttanut sen tarjoama mahdollisuus jakaa helposti ja nopeasti sisältöä ja palveluita. Samanaikaisesti myös käyttäjämäärät ovat kasvaneet eksponentiaalisesti. Tämä onkin aiheuttanut suuria vaatimuksia Internetin kaistanleveydelle sekä palvelimille, joiden avulla sisältöä ja palveluita tarjotaan. Käyttäjämäärien räjähdysmäisen kasvun vuoksi monet verkkopalvelut ovatkin olleet kykenemättömiä vastaamaan tähän tarpeeseen ja verkkopalvelun toiminnot ovat saattaneet hidastua radikaalisti. Näiden ongelmien ratkaisemiseksi on kehitetty erilaisia sisällönjakeluverkkoja (engl. Content Delivery Networks). (Buyya et al., 2008, s. vii).

Sisällönjakeluverkko on kokoelma ympäri internetiä sijoitettuja yhteistyössä toimivia verkkoelementtejä, joiden avulla sisältö replikoidaan useille peilipalvelimille. Peilipalvelinten avulla sisältö saadaan toimitettua näkymättömästi ja tehokkaasti loppukäyttäjälle. Sisällönjakeluverkot ovat syntyneet tarpeesta selvitä luontaisista internetin rajoituksista sekä tarjota loppukäyttäjälle parempi käyttökokemus (engl. QoS, Quality of Service) verkkopalveluiden käyttämisessä. Sisällönhallintaverkkojen tarjoamien palveluiden avulla verkon suorituskykyä saadaan parannettua maksimoimalla kaistanleveys, parantamalla saavutettavuutta sekä ylläpitämällä tiedon oikeellisuutta tiedon replikoinnissa. (Buyya et al., 2008, s. 3-4).

Buyya et al. (2008, s. 4) listaavat tyypillisimmiksi sisällönjakeluverkkojen toiminnallisuuksiksi seuraavat:

- *Uudelleenohjauspyynnöt ja sisällönvälityspalvelut* - pyynnön ohjaaminen lähimmälle soveltuvalle sisällönjakeluverkon välimuistipalvelimelle käyttäen mekanismeja, joilla vältetään tungokset samalle välimuistipalvelimelle.

- *Sisällön ulkoistus- ja jakelupalvelut* - sisällön replikointiin ja/tai välimuistiin hakemiseen sisällön lähdepalvelimelta.

- *Sisällön neuvottelupalvelut* - käyttötarpeiden yksilöintiin yksilö- tai ryhmätasolla.

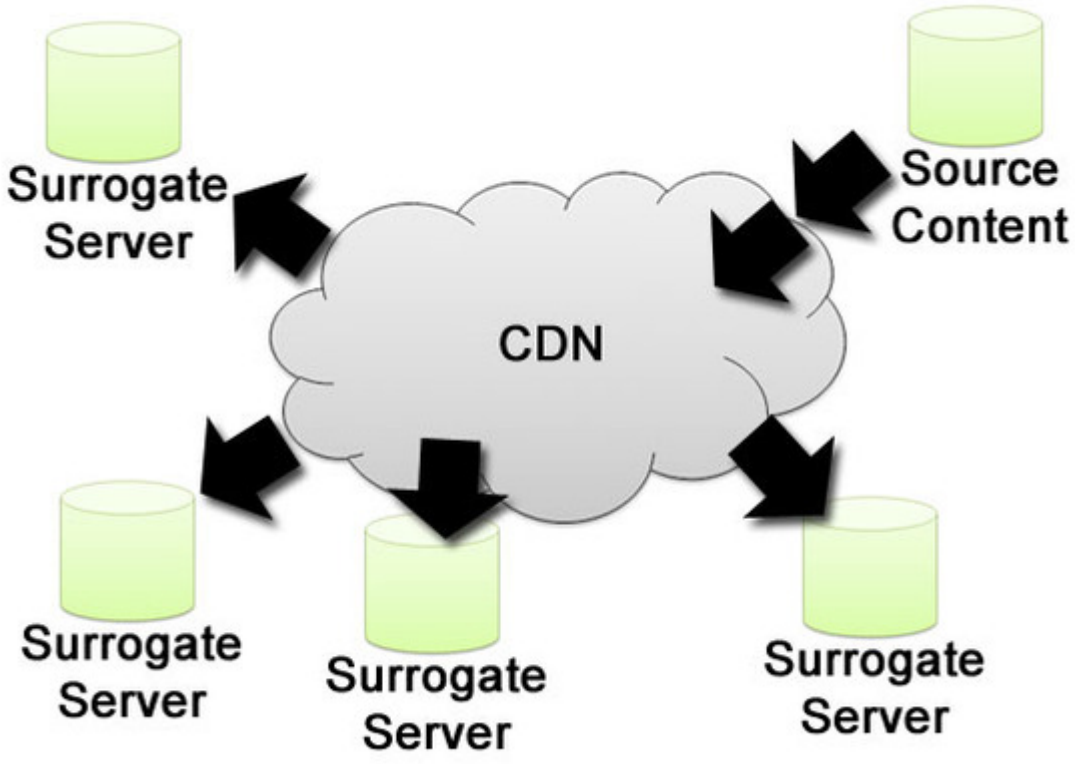
- *Hallintapalvelut* - verkkokomponenttien ja käyttäjätilien hallintaan sekä sisällön käytön valvontaan ja raportointiin.

3.2 Sisällönjakeluverkkojen toimintaperiaatteet

Sisällönjakeluverkkojen toiminta rakentuu useista eri komponenteista. Tässä luvussa esitellään tarkemmin sisällönjakeluverkon komponentteja sekä kyselyn reitittymistä sisällönjakeluverkon sisällä.

3.2.1 Surrogaattipalvelimet

Surrogaaattipalvelimien tarkoitus on keventää lähdepalvelimen kuormaa ottamalla vastuu tiedonvälityksestä loppukäyttäjälle. Tällöin loppukäyttäjälle toimitettava sisältö tulee joko täysin tai osittain surrogaattipalvelimelta, riippuen toimitettavasta materiaalista. Palvelun laadun takaamiseksi (QoS, engl. Quality-of-Service) sisällönjakeluverkon ylläpitäjän on pidettävä huolta siitä, että surrogaattipalvelimet on sijoitettu strategisesti oikein ympäri verkkoa. Tähän tarkoitukseen on kehitetty myös algoritmeja, joiden avulla surrogaattipalvelimet voidaan sijoittaa niin, että suorituskyky kasvaa, mutta infrastruktuurin kulut pysyvät pieninä. (Vakali & Pallis, 2003, s. 69). Kuvassa 3.1 on esitetty surrogaattipalvelinten rooli sisällönjakeluverkossa.



Kuva 3.1: Surrogaattipalvelimet sisällönjakeluverkossa (Ecommerce Developer, 2011)

Toinen ylläpitäjien haaste on määrittää tarvittavien surrogaattipalvelinten lukumäärä. Vakali & Pallis (2003, s. 69-70) jatkavat, että optimaalisen palvelinmäärän selvittämiseen on olemassa kaksi tyypillistä tapaa:

- Yksittäisen palveluntarjoajan lähestymistavassa palveluntarjoajan oman verkon reunoille asetetaan vähintään 40 surrogaattipalvelinta. Maailmanlaajuinen palveluntarjoaja voi näin saavuttaa riittävän maantieteellisen kattavuuden ilman, että se tarvitsee muiden palveluntarjoajien apua. Tämän lähestymistavan heikkoutena saattaa olla se, että surrogaattipalvelimen ja loppukäyttäjän välinen matka saattaa olla joissakin tapauksissa hyvinkin pitkä.

- Useamman palveluntarjoajan lähestymistavassa surrogaattipalvelimia sijoitetaan niin monen globaalin palveluntarjoajan verkkoon kuin mahdollista. Tavoitteena on saada surrogaattipalvelin mahdollisimman lähelle loppukäyttäjää. Optimaalisessa tilanteessa surrogaattipalvelin sijaitsee saman palveluntarjoajan verkossa kuin loppukäyttäjäkin. Tällöin verkon yhteyspisteiden lukumäärä saadaan minimoitua. Tämän lähestymistavan heikkoudeksi puolestaan saattavat muodostua järjestelmän monimutkaisuus sekä tästä johtuva vaikea ylläpidettävyys.

3.2.2 Nimipalvelinkyselypohjainen kuormantasaus

DNS (engl. Domain Name System) on internetin nimipalvelujärjestelmä. Internetissä olevat tietokoneet kommunikoivat keskenään numeeristen osoitteiden avulla, joiden muistaminen olisi ihmiselle liian työlästä. Nimipalvelujärjestelmän avulla numeeriset osoitteet on korvattu helpommin muistettavilla nimillä. Nimipalvelimien avulla nimet voidaan jälleen muuntaa takaisin IP-osoitteiksi, jotta yhteyden muodostaminen kohdekoneelle onnistuisi. Mikäli sillä nimipalvelimella ei ole tiedossaan IP-osoitetta pyydetylle kohdenimelle, jolta osoitetta kysyttiin, niin tämä nimipalvelin välittää kyselyn verkossa eteenpäin kyseisestä osoitteesta vastaavalle nimipalvelimelle. Yksi nimipalvelinkysely saattaa siis kulkea useankin nimipalvelimen kautta ennen kuin lopullinen vastaus saadaan. (Vrt. Liu & Albitz, 2006; Vrt. Mockapetris, P., 1987).

Sisällönjakeluverkkojen yhteydessä nimipalvelimilla on myös tärkeä rooli. Kuvassa 3.2 on esitetty nimipalvelinkyselyn reitittyminen lähimmän surrogaattipalvelimen päättelyssä. Vakali & Pallis (2003, s. 70-71) jakavat oikean surrogaattipalvelimen paikallistamisen viiteen eri vaiheeseen:

1. Käyttäjä lähettää nimikyselyn paikalliselle nimipalvelimelle, joka ohjaa kyselyn eteenpäin erilliseen sisällönjakeluverkon pyynnön ohjaus -infrastruktuuriin (RRI, engl. request-routing infrastructure).

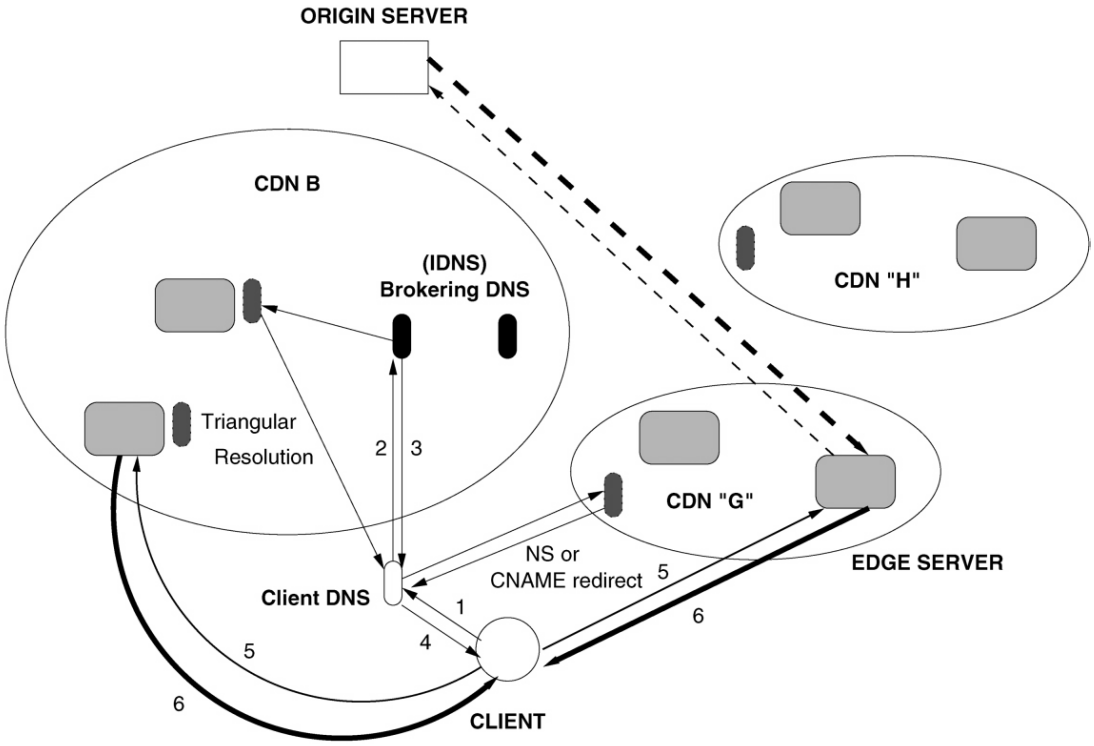
2. RRI kysyy jokaiselta surrogaattipalvelimelta niiden yksityiskohtaista reittiä tälle paikalliselle nimipalvelimelle.

3. Jokainen surrogaattipalvelin lähettää mittaustuloksensa sekä RRI:lle että paikalliselle nimipalvelimelle. RRI:lle lähetetään lisäksi muita RRI:n päätökses-

sään tarvitsemia tietoja, kuten latenssi, hukkaan menneet paketit, reitillä olevien koneiden lukumäärä sekä tietoa surrogaattipalvelinten kuormituksesta.

4. RRI vertailee kaikkia aikaisempia mittaustuloksia ja valitsee sopivimman surrogaattipalvelimen sekä välittää tämän surrogaattipalvelimen IP-osoitteen käyttäjän paikalliselle nimipalvelimelle.

5. Paikallinen nimipalvelin välittää saamansa IP-osoitteen käyttäjälle.



Kuva 3.2: Nimipalvelinkyselyn reitittyminen sisällönjakeluverkossa (Biliris et al., 2002)

3.2.3 Sisällön ulkoistaminen

Kun surrogaattipalvelimet on sijoitettu sisällönjakeluverkkoon strategisesti oikeisiin sijainteihin, tulee seuraavaksi valita tapa välittää sisältö lähdepalvelimelta surrogaattipalvelimille. Kirjallisuudesta löytyy tähän kolme erilaista tapaa:

1. *Kooperatiivinen työntöpohjainen lähestymistapa* (engl. cooperative push-based approach): Lähdepalvelin lähettää sisällön itse surrogaattipalvelimelle. Kyselyt ohjataan loppukäyttäjää lähimpänä olevalle surrogaattipalvelimelle tai lähdepalvelimelle. (Xu et al., 2006).

2. *Ei-kooperatiivinen vetopohjainen lähestymistapa* (engl. non-cooperative pull-based approach): Asiakkaan pyyntö ohjataan DNS-pohjaisella reitityksellä lähimmälle surrogaattipalvelimelle. Jos tiedostoa ei löydy surrogaattipalvelimen

välimuistista, niin surrogaattipalvelin noutaa ensin tiedoston lähdepalvelimelta. (Dilley et al., 2002).

3. *Kooperatiivinen vetopohjainen lähestymistapa* (engl. cooperative pull-based approach): Menetelmä on muuten samanlainen kuin edellinen, mutta surrogaattipalvelimet pystyvät kommunikoimaan keskenään. Tällöin puuttuvan tiedoston tapauksessa surrogaattipalvelin ei noudakaan puuttuvaa tiedostoa lähdepalvelimelta, vaan se selvittää lähimmän surrogaattipalvelimen, jolta kyseinen tiedosto on saatavilla. (Xu et al., 2006).

3.2.4 Tilinhallinta- ja laskutusmekanismit

Laajan sisällönjakeluverkon rakentaminen ja ylläpitäminen vaativat paljon resursseja. Harvalla organisaatiolla on perusteltu syy rakentaa oma sisällönjakeluverkko, vaikka tarve sellaiselle olisi. Tähän tarpeeseen vastaavat palveluntarjoajat, jotka myyvät käyttöoikeuksia omaan sisällönjakeluverkkoonsa. Tällöin on teknisesti erotettava asiakkaat toisistaan, jotta asiakkaiden laskutus käyttömäärien perusteella on mahdollista.

Palveluntarjoajat laskuttavat asiakkaitaan surrogaattipalvelinten tiedonsiirtomäärien perusteella. Sisällönjakeluverkkopalveluista laskuttamisessa on kuitenkin paljon erilaisia haasteita. Loppukäyttäjälle määräytyvä hinta koostuu monesta eri komponentista. Siihen vaikuttavat muun muassa siirtokaistasta ja -määrästä maksettava hinta, surrogaattipalvelimille hajautettujen tiedostojen koko, käytössä olevien surrogaattipalvelinten lukumäärä, käyttövarmuuden ja vakauden takaaminen sekä tietoturva-asetukset sisällön replikoinnissa. (Vrt. Krishnamurthy et al., 2001).

3.3 Tavanomaiset sisällönjakeluverkkojen arkkitehtuurit

Sisällönjakeluverkkojen arkkitehtuurit jaetaan tavanomaisesti kahteen pääluokkaan: palvelin-asiakas-arkkitehtuuriin sekä vertaisverkkoarkkitehtuuriin. Verrattaessa arkkitehtuurimalleja keskenään, havaitaan palvelin-asiakas-arkkitehtuurimallin olevan selkeästi kaupallisempi. Palvelin-asiakas-mallin mukaisen sisällönjakeluverkon ylläpitäminen ja käyttäminen on kallista. Vertaisverkkoihin perustuva arkkitehtuurimalli puolestaan pyrkii minimoimaan kulut poistamalla tarpeen keskitetyille palvelimille, mutta sen toimivuus ja luotettavuus riippuu suoraan aktiivisten käyttäjien määrästä. (Vrt. Mulerikkal & Khalil, 2007).

Myös näiden kahden mallin yhdistäminen on yleistä. Esimerkiksi musiikkipalvelu Spotify[5] hyödyntää vahvasti vertaisverkkomallia musiikin välittämisessä

[5] `http://pansentient.com/2011/04/spotify-technology-some-stats-and-how-spotify-works/`

loppukäyttäjälle. Vain murto-osa todellisesta kuunnellusta musiikista tulee suoratoistodatana Spotify-palvelun omilta palvelimilta.

3.3.1 Palvelin-asiakas-arkkitehtuuri

Kaupallisella arkkitehtuurilla tarkoitetaan yleensä palvelin-asiakas-arkkitehtuurimallin mukaista toteutusta. Tyypillinen esimerkki kaupallisesta sisällönjakeluverkosta on Akamai[6]. Akamain keskitetystä lähestymistavassa asiakkaat vuokraavat tilaa Akamain sisällönjakeluverkon palvelimilta. Akamain sisällönjakeluverkko koostuu kymmenistä tuhansista surrogaattipalvelimista. (Mulerikkal & Khalil, 2007, s. 360).

3.3.2 Vertaisverkkoarkkitehtuuri

Vertaisverkossa (engl. peer-to-peer) kaikki toisiinsa yhteydessä olevat koneet voidaan nähdä yhtenä isona resurssina, josta on mahdollista etsiä ja noutaa tietoa. Vertaisverkossa ei ole erillisiä palvelimia ja asiakkaita, vaan kaikki vertaisverkon koneet toimivat sekä palvelimina että asiakkaina. Sen sijaan, että sisältö olisi delegoitu yksittäiselle taholle, kuten Akamaille, voivat koneet jakaa paikallista (halpaa) resurssia ja saada vastineeksi (arvokasta) ulkoista resurssia. (Mulerikkal & Khalil, 2007, s. 360).

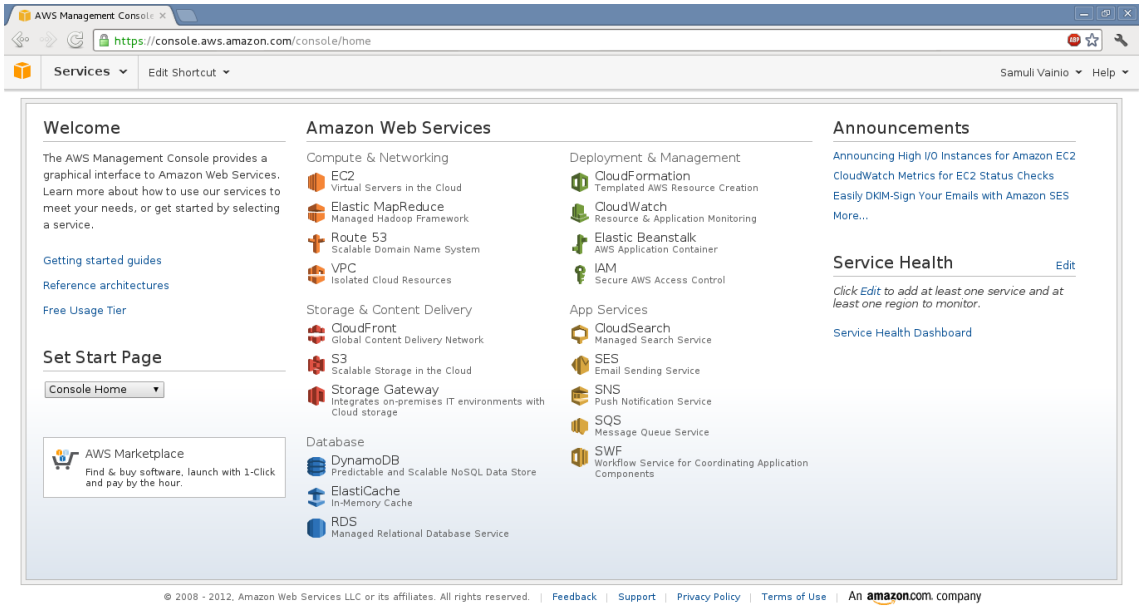
[6] `http://www.akamai.com/`

4 Amazonin pilvipalvelutarjonta

Tässä luvussa esitellään kattavasti Amazon Web Services -palvelualustan tarjoamia sovelluksia. Luvussa 5 on esitetty varsinainen Amazonin pilvipalveluiden käyttöönotto. Pilvipalvelutarjontaa esitetään tarkasti vain siltä osin kuin ne liittyvät luvussa 6 toteutettavan Drupal-järjestelmän tarpeisiin.

4.1 Työn kannalta keskeisimmät palvelut

Amazonin tarjoamien palveluiden käyttöönotto ja hallinta tapahtuu graafisen käyttöliittymän, AWS-hallintakonsolin[7] (engl. Amazon Web Services Management Console) avulla. Kuvassa 4.1 on esitetty AWS-hallintakonsolin etusivu. Monille palveluille tämä hallintakonsoli on ainut tapa kyseisen palvelun hallintaan.



Kuva 4.1: Amazon Web Services -hallintapaneelin etusivu

Joidenkin palveluiden hallintaan on myös olemassa komentorivipohjaisia työkaluja. Tässä työssä niistä käytetään AutoScaling[8]- ja CloudWatch[9]-työkaluja. Nämä työkalut tarvitsevat toimiakseen tunnuskohtaisia, salaisia avaimia, joiden avulla tunnistautuminen tapahtuu. Nämä tunnukset ovat jokaisen käyttäjän saatavilla hallintapaneelin Oma tili -valikon kautta.

[7] `https://console.aws.amazon.com/console/home/`
[8] `http://aws.amazon.com/developertools/2535`
[9] `http://aws.amazon.com/developertools/2534`

4.1.1 Amazon EC2

Amazon EC2[10] (engl. *Elastic Compute Cloud*) on elastinen pilvialusta, joka mahdollistaa skaalautuvan pilvilaskentaympäristön luonnin. Palvelun web-käyttöliittymän avulla on helppo luoda uusia palvelininstansseja käyttäen jotakin tarjolla olevaa AMI-levykuvaa (engl. Amazon Machine Image). Amazon tarjoaa itse useita omia levykuviaan, jotka on optimoitu pilvilaskentakäyttöön. Levykuvan voi myös valita yhteisön kautta tarjolla olevista levykuvista tai omista tallennetuista levykuvista. Käyttöjärjestelmävaihtoehtoja on useita ja niiden määrä kasvaa koko ajan. Amazonin omasta käyttöjärjestelmävalikoimasta löytyvät useat erilaiset Linux-jakeluversiot sekä Windows Server -käyttöjärjestelmät. Käyttöjärjestelmän voi asentaa myös tiettyyn käyttötarkoitukseen tarkoitettujen sovellusten kanssa. Palvelimesta voivat löytyä esiasennettuina esimerkiksi tietokanta- ja web-palvelinsovellukset. Amazon Marketplace tarjoaa lisäksi laajan valikoiman sekä ilmaisia että maksullisia ohjelmistoja, jotka voi helposti asentaa levykuvan mukana uuteen järjestelmään.

Tarjolla olevat EC2-instanssityypit jaotellaan kahteen luokkaan: ensimmäisen ja toisen sukupolven instansseihin. Ensimmäisen sukupolven instanssit tarjoavat kustannustehokkaan alustan, joka soveltuu moniin käyttötarkoituksiin. Ne käyttävät paikallisia kiintolevyjä ja niiden laskentatehoa voidaan nostaa enintään kahdeksaan laskentayksikköön instanssia kohden. Toisen sukupolven instanssit puolestaan käyttävät järjestelmälevynä vain ja ainoastaan suorituskykyisempää verkkolevytyyppistä EBS-levyjärjestelmää[11] (engl. Elastic Block Store) sekä tukevat useampaa laskentayksikköä instanssia kohti kuin ensimmäisen sukupolven instanssit. Tarkempi luokittelu tapahtuu instanssille varattujen resurssien perusteella. Riippuen käyttötarkoituksesta, voi instanssilla olla esimerkiksi paljon prosessoritehoa, keskusmuistia tai levyn suorituskykyä.

EBS-levyt näkyvät ja toimivat järjestelmässä samoin kuin paikalliset levytkin. Toisin kuin tavalliset kiintolevyt, ovat EBS-levyt automaattisesti kahdennettu. Tällöin tallennettua tietoa ei katoa yksittäisten laitteistokomponenttien rikkoutuessa. EBS-levyn tyypiksi on mahdollista valita Provisioned IOPS, jolla voi taata levyn suorituskyvyn. Oletuksena EBS-levy pystyy käsittelemään 100 I/O-operaatiota sekunnissa. Provisioned IOPS -työkalun avulla tuettavien levyoperaatioiden määrää voidaan nostaa jopa 2 000 I/O-operaatioon sekuntia kohti.

Kuormantasaajan[12] avulla sisään tulevan liikenteen voi ohjata tasapuolisesti käytössä oleville EC2-instansseille. Sen avulla on mahdollista saavuttaa entistäkin suurempi järjestelmän vikasietoisuus. Kuormantasaaja tunnistaa automaattisesti epänormaalisti toimivat instanssit ja uudelleenohjaa liikenteen toimiville instans-

[10] `http://aws.amazon.com/ec2/`
[11] `http://aws.amazon.com/ebs/`
[12] `http://aws.amazon.com/elasticloadbalancing/`

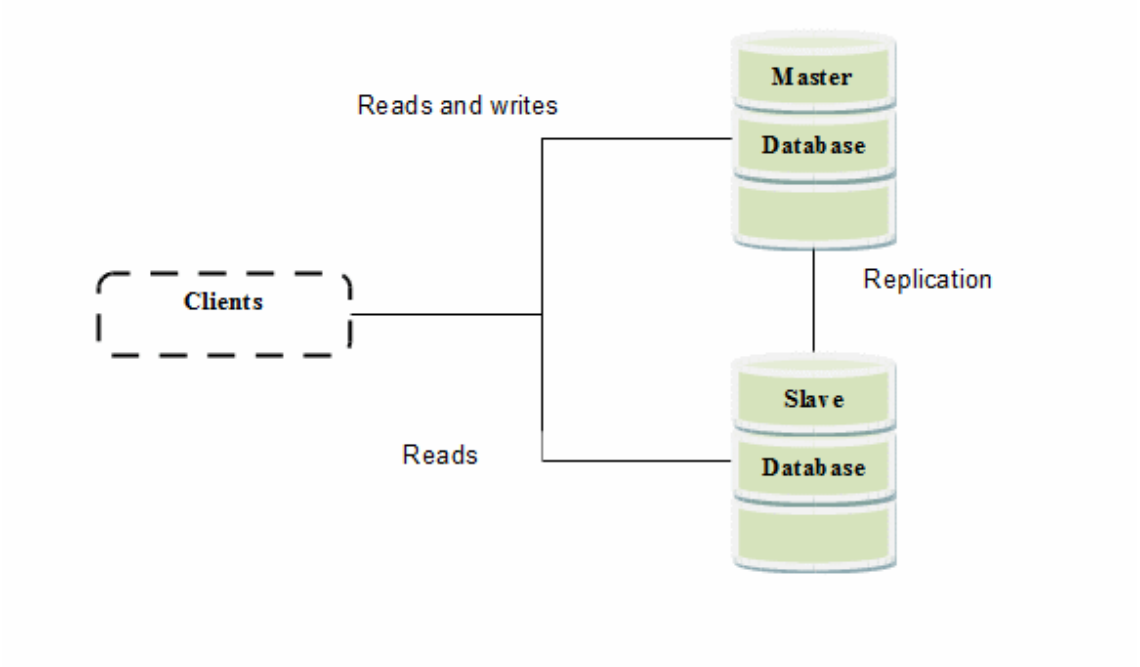
seille siihen asti, että tämä instanssi taas toimii.

4.1.2 Amazon RDS

Amazon RDS[13] (engl. *Relational Database Service*) mahdollistaa relaatiotietokantojen luomisen pilviympäristöön. Web-käyttöliittymän avulla on mahdollista luoda, hallita ja skaalata relaatiotietokantoja. Palvelusta löytyy tuki yleisimmille tietokantajärjestelmille. Tällä hetkellä tuettuja tietokantajärjestelmiä ovat MySQL, Oracle sekä Microsoft SQL Server. RDS tarjoaa kustannustehokkaan tavan siirtää vastuu tietokantapalvelinten ylläpidosta Amazonille. RDS hoitaa automaattisesti tietokannan varmuuskopiot, tietoturvapäivitykset sekä mahdollisen replikoinnin.

RDS tarjoaa myös Provisioned IOPS -toiminnallisuuden, jonka avulla tuettujen I/O-kyselyiden määrä voidaan nostaa jopa 10 000 kyselyyn sekunnissa. Provisioned IOPS -toiminnallisuuden ja master-slave-arkkitehtuurin avulla tietokantajärjestelmä saadaan siis skaalautumaan tarpeiden mukaisesti. Hendersonin (2006) mukaan master-slave-malli on yksi yleisimmin käytetyistä arkkitehtuurimalleista tietokantojen replikoinnissa. Kuvassa 4.2 on esitetty master-slave-arkkitehtuurin mukainen toteutus. Master-palvelin käsittelee jokaisen kirjoituskyselyn ja lukukyselyt ohjataan vain luku -tilassa oleville slave-palvelimille, eli replikoille. Myös master-palvelin vastaa lukukyselyihin. Yhtä master-palvelinta kohti voi olla useampia slave-palvelimia, jotka ovat ajantasaisia kopioita master-palvelimen sisällöstä. Tämä arkkitehtuuri tukee erityisesti web-sivustoja, joissa on paljon lukukyselyitä, mutta vähän kirjoituskyselyitä. Lisäksi arkkitehtuuri mahdollistaa slave-palvelimen siirtymisen master-palvelimeksi, mikäli master-palvelin rikkoutuu yllättäen.

[13] `http://aws.amazon.com/rds/`



Kuva 4.2: Master-slave-arkkitehtuurin mukainen replikointi (Henderson, 2006, s. 233)

Amazonin palvelinkeskusten sisällä on erillisiä saatavuusalueita (engl. availability zone), eli toisistaan eristettyjä palvelinsaleja. Esimerkiksi Irlannin palvelinkeskus on jaettu kolmeen saatavuusalueeseen. Nämä palvelinkeskukset ovat nopeassa verkkoyhteydessä keskenään ja tarjoavat mahdollisuuden hajauttaa kriittiset tietokantapalvelut useampaan saatavuusalueeseen. Jos yksittäisessä palvelinsalissa sattuu jokin ennalta-arvaamaton vikatilanne, kuten pitkä sähkökatkos, niin tällöin on mahdollista siirtyä automaattisesti käyttämään toisessa palvelinsalissa sijaitsevaa replikaa. RDS:n Multi-AZ-toiminnallisuus mahdollistaa juuri tällaisiin tilanteisiin varautumisen. Multi-AZ-tilassa toimiva tietokantapalvelin reagoi oletuksena käytettävän saatavuusalueen virhetilanteeseen kohdistamalla tietokantakyselyt toisen saatavuusalueen tietokantaan. Optimitilanteessa koko palvelinkeskuksen toimintahäiriö ei siis edes näy loppukäyttäjälle ollenkaan.

4.1.3 Amazon S3

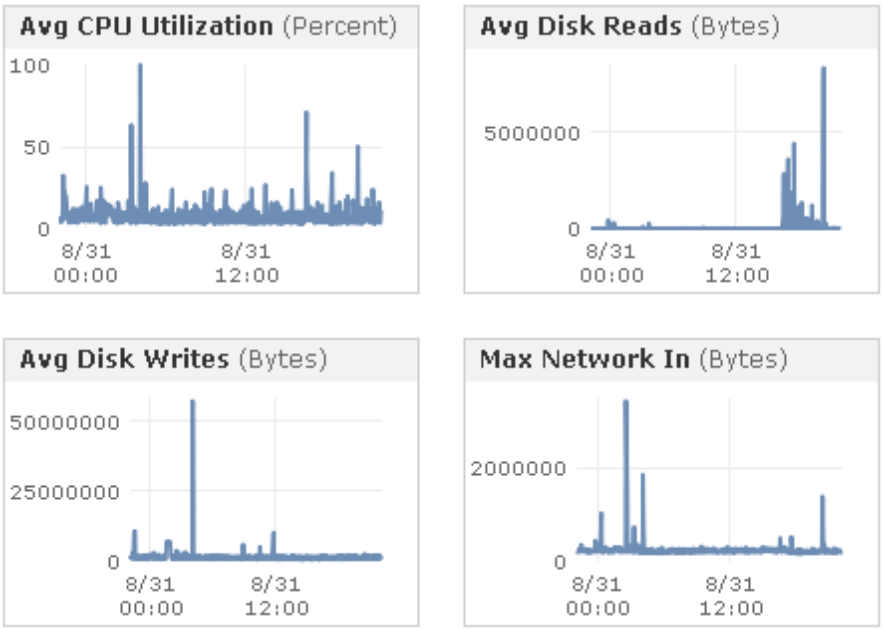
Amazon S3 (engl. *Simple Storage Service*) tarjoaa menetelmän tallettaa ja jakaa tietoa web-rajapinnan yli. S3 koostuu säiliöistä (engl. bucket), Järjestelmä tukee REST- ja SOAP-rajapintoja, joiden avulla S3-levyjärjestelmä voidaan helposti integroida moneen eri käyttötapaukseen ja erilaisille päätelaitteille. Palvelun avulla on mahdollista siirtää kaikki järjestelmän tietosisältö pilveen. Näin voidaan hyödyntää pilven automaattinen skaalautuvuus sekä käyttömääräperustainen laskutus.

Avoimen rajapinnan avulla S3-levyjärjestelmä on mahdollista liittää käyttöjärjestelmään ja saada se toimimaan kuin paikallinen kiintolevy. Tämä onnistuu

esimerkiksi Linux-käyttöjärjestelmissä s3fs[14]-työkalun avulla. Tällöin sama tietovarasto voidaan saumattomasti liittää samanaikaisesti useampaankin palvelimeen. Tätä ominaisuutta käsitellään tarkemmin luvussa 6.2.

4.1.4 Amazon CloudWatch

Amazon CloudWatch tarjoaa monipuolisen monitorointityökalun palvelininstanssien ja levytilan seurantaan. CloudWatch seuraa ja tallettaa instanssikohtaisesti useita erilaisia mitattavia arvoja. Tällaisia arvoja ovat muun muassa palvelimen latenssi, prosessorin ja muistin käyttöaste tai levyn käyttö. Palvelun käyttöönotto ja tiedonkeruu ei vaadi erillisen ohjelmiston asentamista. Työkalu luo erilaisia kuvaajia kerätyn tiedon pohjalta. Kuvassa 4.3 on esitetty keskimääräinen prosessorin käyttö, levyn kirjoitus- ja lukukyselyiden määrä sekä verkkoliikenteen määrä.



Kuva 4.3: CloudWatch-monitorointityökalun kuvaajia (Amazon Web Services Blog, 2009)

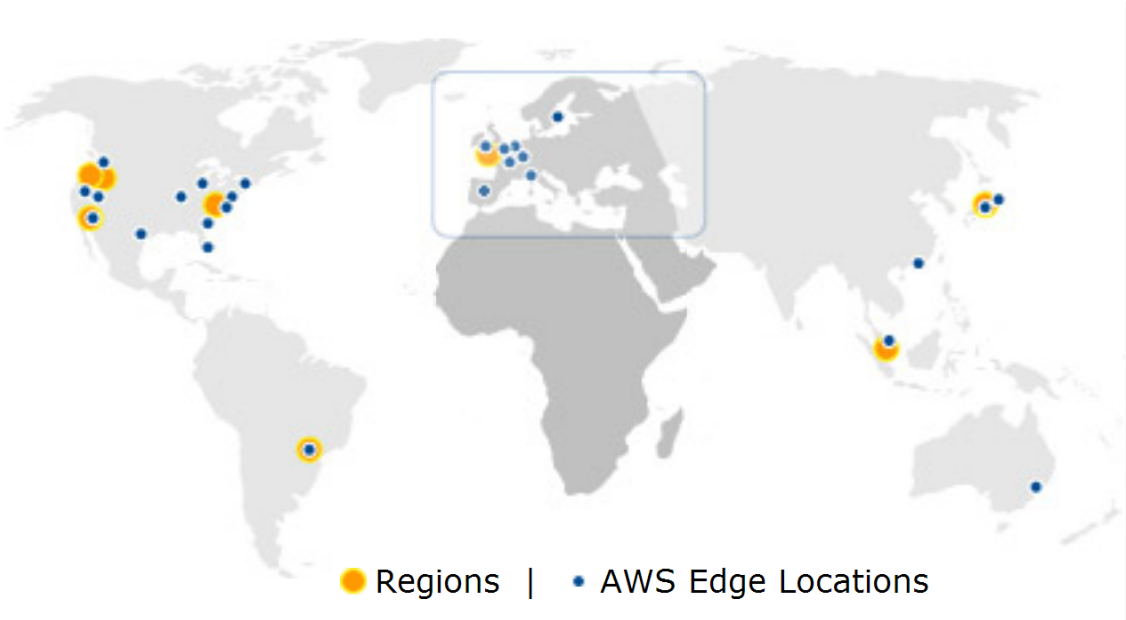
CloudWatch-työkalun keräämää informaatiota on mahdollista myös tarkkailla ohjelmistollisesti. Mittausarvojen seuranta on ensisijaisen tärkeää, kun halutaan varmistaa palvelun laatu. Tällöin on reagoitava tilanteisiin, joissa jokin arvo nousee yli hälytysrajan. Esimerkiksi prosessorin käyttöaste voi olla liian korkea. Tällaisissa tilanteissa palvelu voi esimerkiksi lähettää ylläpitäjälle sähköpostiviestinä varoituksen, että hälytysraja on ylitetty. Tämä vaatisi kuitenkin ympärivuorokautista ylläpitohenkilökuntaa ja se ei olisi erityisen kustannustehokasta. Ratkaisuna tähän CloudWatch tarjoaakin automatisoituja toimenpiteitä hälytyksen tullessa.

[14] `http://code.google.com/p/s3fs/wiki/FuseOverAmazon`

Monitorointityökaluun voidaan määrittää erilaisia laukaisimia (engl. triggers) sekä toimenpiteitä (engl. actions). Jokaiselle mitattavalle suureelle voidaan määrittää useitakin eri laukaisimia ja toimenpiteitä. Esimerkiksi prosessorin käyttöasteelle voidaan asettaa laukaisin, joka laukeaa kun prosessorin keskimääräinen käyttöaste on yli 50 %. Tällöin suoritetaan jokin ennalta määritetty toimenpide. Tällaisessa tilanteessa tämä toimenpide voisi olla esimerkiksi uuden palvelininstanssin käynnistäminen ja liittäminen kuormantasaajaan. Nyt kun pilvessä on uusi palvelin, niin rasitus jakautuu tasaisesti myös sille ja keskimääräinen prosessorin käyttöaste pienenee. Jos käyttöaste on kuitenkin edelleen yli hälytysrajan, syntyy uusi hälytys. Tällöin sama toimenpide suoritetaan uudelleen niin kauan, että käyttöaste saadaan tippumaan alle hälytysrajan. Rasituspiikin poistuessa pilvessä voi olla hälytysten jälkeen huomattavasti enemmän laskentakapasiteettia kuin on tarve. Tähän voidaan reagoida esimerkiksi alhaisen käyttöasteen hälytyksellä. Nyt pilvestä poistetaan instansseja, kun keskimääräinen käyttöaste on esimerkiksi alle 10 %. Näin saavutetaan kustannustehokas automaattinen skaalautuvuus.

4.1.5 Amazon CloudFront

Amazon CloudFront on Amazonin infrastruktuuriin integroitu sisällönjakeluverkko. CloudFront-palvelun avulla on mahdollista jakaa loppukäyttäjälle staattista, dynaamista tai suoratoistodataa (engl. streaming) loppukäyttäjää lähimpänä olevasta sisällönjakeluverkon reunasijainnista (engl. edge location). Kuvassa 4.4 on esitetty Amazonin infrastruktuuri sisällönjakeluverkon osalta. Keltaiset pallot kuvaavat palvelinkeskusten sijaintia ja siniset sisällönjakeluverkon reunasijainteja.



Kuva 4.4: Amazon CloudFront -sisällönjakeluverkon reunasijainnit

4.2 Palvelinkeskusten sijainnit

Linthicumin (2006, s. 226-227) mukaan kaksi tärkeintä tekijää pilven suorituskyvyn kannalta ovat alustan suorituskyky sekä verkon latenssi. Verkon latenssin suhteen loppukäyttäjän maantieteellinen sijainti on ratkaisevassa osassa. Kuten kuva 4.4 osoitti, niin Amazonin sisällönjakeluverkossa lähimpään reunapalvelimeen saattaa esimerkiksi Aasiassa olla matkaa useita tuhansia kilometrejä. Tämä saattaa muodostua ongelmaksi näissä tietyissä maantieteellisissä sijainneissa sellaisten sovellusten kanssa, jotka vaativat erittäin nopean verkkoyhteyden palvelimeen. Taulukossa 4.1 on esitetty verkon latensseja Suomesta Amazonin eri palvelinkeskuksiin. Mittaukset on suoritettu Tampereelta ja viitearvona on myös esitetty verkon latenssi Helsinkiin.

Taulukko 4.1: Latenssi Suomesta Amazonin pilvipalvelinkeskuksiin

Palveluntarjoaja	**Keskimääräinen latenssi**
FUNET, Helsinki	3.36 ms
Amazon EU (Irlanti)	55.13 ms
Amazon US / East (Virginia)	120.59 ms
Amazon US / West (California)	181.39 ms
Amazon US / West (Oregon)	188.76 ms
Amazon South America (São Paulo)	255.09 ms
Amazon Asia Pacific (Tokyo)	300.83 ms
Amazon Asia Pacific (Singapore)	362.18 ms

Maantieteellisen sijainnin ja taulukon 4.1 avulla nähdään selkeästi, että Irlannissa sijaitsevaan Amazonin palvelinkeskukseen on selkeästi muita pienempi latenssi. Se on kuitenkin moninkertaisesti suurempi kuin latenssi Helsinkiin ja tämä näkyy myös kaistanleveydessä. Tämä tarkoittaa sitä, että jos sovelluksen toiminnalle on asetettu latenssivaatimuksia, esimerkiksi alle 30 ms, niin tällöin Amazonin palveluiden käyttö ei ole mahdollista. Esimerkiksi verkkopelien toimivuus on yleensä sitä parempi, mitä pienempi vasteaika on.

4.3 Käyttökustannukset

Pilvipalveluntarjoajien hinnoittelumallit poikkeavat huomattavasti tavanomaisesta kuukausimaksupohjaisesta laskutuksesta. Esimerkiksi OVH[15] tarjoaa dedikoituja palvelimia, joiden kuukausihinta pitää sisällään kaikki palvelusta aiheutuvat kustannukset, myös rajattoman verkkoliikenteen. Pilvipalveluiden tapauksessa sen sijaan kaikki komponentit hinnoitellaan erikseen. Tämän avulla on mahdollista saavuttaa huomattavaakin kustannussäästöä, kun käytettävissä olevaa laskentakapasiteettia voidaan automaattisesti skaalata käyttötarpeen mukaan.

[15] `http://www.ovh-hosting.fi/dedikoidut_palvelimet/`

On kuitenkin selkeästi olemassa skenaarioita, joissa pilvipalveluiden käytöstä syntyy huomattavasti suurempi kuukausilasku. Esimerkiksi Amazonin pilvipalveluissa käytetystä siirtokaistasta laskutetaan $0.12 per siirretty gigatavu. Tiedostonjakopalveluilla siirtomäärät ovat useita teratavuja kuukaudessa. Amazonin hinnaston mukaan tämä tuottaisi kuukausitasolla jo useiden satojen dollareiden laskun. Tällaisessa käyttötarkoituksessa Amazon ja pilvipalvelinratkaisu ei siis välttämättä ole oikea lähestymistapa. Sen sijaan tilanteissa, joissa keskimääräinen laskentatehon tarve on hyvin alhainen, mutta silloin tällöin on tarvetta suurellekin laskentakapasiteetille, niin Amazonin pilvipalvelinratkaisu on huomattavasti kustannustehokkaampi.

4.3.1 Hinnoittelutavat

Amazon tarjoaa EC2-instansseille kolme erilaista hinnoitteluvaihtoehtoa[16]: maksu käytön mukaan (engl. on-demand), varattu instanssi (engl. reserved instance) sekä huutokauppatyyppisesti muodostettava käyttötunnin hinta Amazonin käyttämättömästä laskentainstanssista (engl. spot instance). Jokaisessa hinnoitteluvaihtoehdossa on kuitenkin instansseilla aina tuntipohjainen hinta, joka vaihtelee varattujen resurssien ja käytetyn hinnoittelupolitiikan mukaan.

On-demand-instansseja voi käynnistää ja sammuttaa vapaasti. Lasku muodostuu suoraan käyttötuntien mukaan. Sama hinnoittelumalli pätee myös spot instance -mallissa. Varattujen instanssien suhteen tilanne on kuitenkin toinen. Instanssin voi varata joko yhden tai kolmen vuoden sopimuksella. Varatun instanssin hinta muodostuu etukäteen maksettavasta kiinteästä summasta sekä lisäksi käyttötunneista maksettavasta, huomattavasti on-demand-hintaa pienemmästä tuntihinnasta.

Varatut instanssit jaetaan lisäksi kolmeen tyyppiin: kevyeen, keskikokoiseen ja vaativaan laskentaan. Palvelimen käytössä olevat resurssit ovat kuitenkin kaikissa samat, kyse on vain hinnoitteluerosta. Kevyeen laskentaan tarkoitetun instanssin varaamisessa etukäteismaksu on muita pienempi, mutta tuntihinta on kaikkein suurin. Keskikokoiseen laskentaan soveltuvan instanssin etukäteishinta on puolestaan hieman suurempi, mutta tuntihinta taas selkeästi pienempi. Vaativaan laskentaan käytettävän palvelimen hinnoittelupolitiikka sen sijaan eroaa aiemmin esitellyistä. Kevyeen ja keskikokoiseen laskentaan käytettyjen palvelinten käytöstä maksetaan tuntipohjaisesti, kun taas vaativaan laskentaan tarkoitetusta instanssista joutuu maksamaan tuntihintaa, vaikkei palvelin ole käytössä. Tämä on kuitenkin huomioitu instanssin tuntihinnassa, joka on selkeästi muita vaihtoehtoja edullisempi.

Optimaalisen hinnoittelutavan valinta riippuu siis vahvasti käyttötarpeesta. Hajanaiseen kehityskäyttöön on-demand-tyyppinen hinnoittelutapa saattaa olla huomattavasti muita vaihtoehtoja edullisempi. Pysyvään ja pitkäaikaiseen verkkopalvelun pyörittämiseen sen sijaan on kustannustehokkainta ostaa instanssi usean vuoden

[16] `http://aws.amazon.com/ec2/pricing/`

sopimuksella ja käyttää raskaaseen laskentaan soveltuvaa hinnoittelupolitiikkaa. Oikean vaihtoehdon valinta ilman aikaisempaa käyttöstatistiikkaa voi olla hankalaa, eikä tällöin kannata heti sitoutua monen vuoden kiinteään sopimukseen. Käyttöstatistiikan kerääminen on-demand-tyyppisellä hinnoittelulla voi tuoda kustannussäästöjä, kun käyttötarve saadaan ensin paremmin hahmotettua.

4.3.2 Kustannusarvioita

Amazon tarjoaa työkalun[17], jonka avulla on mahdollista laskea arvio kuukauden aikana syntyvistä kuluista. On kuitenkin tärkeää huomata, että kyseessä on vain arvio, joka perustuu annettuihin arvoihin. Todellisuudessa iso kävijäpiikki voi tuottaa arvioon isonkin poikkeaman.

Valitaan esimerkiksi keskisuuri Drupal-sivusto. Sivuston suunnittelussa halutaan ottaa huomioon vikasietoisuutta sekä mahdollisiin kävijäpiikkeihin halutaan varautua. Tällöin järjestelmän pohjaksi tulee valita kaksi tai useampi EC2-palvelin, jotka on liitetty kuormantasaajaan. Drupal-sivusto vaatii lisäksi tietokantasovelluksen. Vastuu tietokantapalvelimen ylläpidosta halutaan ulkoistaa Amazonille. Taulukossa 4.2 on esitetty tällaiselle Drupal-sivustolle sopiva pilvipalvelinlaitteisto[18].

Taulukko 4.2: Esimerkki keskisuuren Drupal-sivuston pilvipalvelinlaitteistosta

Palvelu	Hinnoittelutapa	Kuvaus
EC2 / M1 Medium Instance (2 kpl)	Vaativa laskenta, kolmen vuoden kiinteä sopimus	2 kpl EC2-laskentayksikköä, 3.75 GB keskusmuistia, 410 GB levytilaa, keskinkertainen I/O-kapasiteetti
RDS / Medium DB Instance (1 kpl)	Vaativa laskenta, kolmen vuoden kiinteä sopimus	2 kpl ECU-laskentayksikköä, 3.75 GB keskusmuistia, keskinkertainen I/O-kapasiteetti
RDS / Small Instance (1 kpl)	Vaativa laskenta, kolmen vuoden kiinteä sopimus	1 kpl ECU-laskentayksikköä, 1.7 GB keskusmuistia, keskinkertainen I/O-kapasiteetti
EC2 / M1 Medium Instance (1-20 kpl)	Maksu käytön mukaan (on-demand)	2 kpl EC2-laskentayksikköä, 3.75 GB keskusmuistia, 410 GB levytilaa, keskinkertainen I/O-kapasiteetti

Järjestelmän pohjana toimivat siis kaksi EC2-palvelinta, jotka ovat aina päällä. Sivuston skaalautuvuus ja kävijäpiikkeihin varautuminen toteutetaan CloudWatch-työkalun avulla. Palvelinten keskimääräisen rasituksen noustessa liian suureksi,

[17] `http://calculator.s3.amazonaws.com/calc5.html`
[18] `http://aws.amazon.com/ec2/instance-types/`

lisätään pilveen laskentatehoa käynnistämällä uusia palvelininstansseja. Tietokantapalvelimelle luodaan lisäksi erillinen vain luku -replika, jolloin master-tietokantapalvelimen kuormaa saadaan kevennettyä.

Esitellyn laitteiston lisäksi kokonaishintaan vaikuttavat monet muutkin tarvittavat komponentit. Muun muassa kuormantasaaja, käytetty levytila sekä verkkoliikenteen määrä ovat erikseen laskutettavia komponentteja. Hintaan vaikuttaa myös käytetty Amazonin pilvipalvelinkeskuksen sijainti. Tässä esimerkissä käytämme Irlannin palvelinkeskusta. Taulukossa 4.3 on esitetty hinta-arvio aikaisemmin esitetylle laitteistolle ja oheiskustannuksille. Komponenttien kokonaishinnaksi esitetyllä kokoonpanolla muodostuu siis noin $175 / kk, alv 0 %.

Taulukko 4.3: Drupal-sivuston arvioitu kuukausilasku

Komponentti	**Käyttö**	**Hinta**
EC2 / M1 Medium (kiinteä, 2 kpl)	2 * 720 h	$0.04/h * 1440 h = $57,60
EC2 / M1 Medium (on-demand)	~30 h	$0.23/h * 30 h = $6,90
RDS / Medium DB (kiinteä)	720 h	$0,059 / h * 720 h = $42,48
RDS / Small DB (kiinteä)	720 h	$0.030/h * 720 h = $21,60
Levytila	100 GB	$0.095/GB * 100 GB = $9,50
Kuormantasaaja	720 h + 100 GB	$0.028/h * 720 h + $0.008/GB * 100 GB = $20,96
Verkkoliikenne (EC2 + CDN)	50 + 50 GB	$0.120/GB * 100 GB = $12,00

Taulukossa esitettyjen hintojen lisäksi tulee huomioida kiinteiden resurssien aloituskustannukset. Kolmen vuoden sopimuksella kahden EC2-instanssin etukäteismaksu on $600 / kpl ja RDS-instanssien $600 + $300, eli yhteensä $2100. Taulukkoon kirjattujen kulujen lisäksi myös levyn I/O-käytöstä syntyy pieni kuluerä. Lisäksi varatusta, mutta ei käytössä olevasta IP-osoitteesta peritään $0,005 / h.

5 Pilvipalvelinjärjestelmän käyttöönotto

Tässä luvussa esitellään pilvipalvelinympäristön käyttöönotto Amazon Web Services -palvelualustan avulla. Tavoitteena on luoda palvelinympäristö, joka soveltuu luvussa 6 käyttöönotettavan Drupal-järjestelmän pohjaksi. Pilvipalvelinympäristön luonnissa kiinnitetään erityistä huomiota skaalautuvuuden tukemiseen.

5.1 EC2-palvelin

Ensimmäinen vaihe palvelinympäristön luomisessa on uuden palvelininstanssin luonti sekä käyttöjärjestelmän asennus. Amazon EC2 (engl. Elastic Compute Cloud) tarjoaa valmiita levykuvia suosituimmista käyttöjärjestelmistä sekä omia erikseen pilvipalvelinympäristöön optimoituja Linux-levykuvia. Yhteisöllä on myös mahdollisuus luoda omia levykuvia ja tehdä näistä julkisia, eli vapaasti kaikkien käytettävissä olevia. Järjestelmään voi myös luoda omia, henkilökohtaisia levykuvia, joista vain kyseisellä käyttäjällä itsellään on oikeus luoda uusia palvelininstansseja. Oletuksena uusissa EC2-instansseissa ei käytetä perinteistä käyttäjänimeen ja salasanaan perustuvaa tunnistautumista, vaan sisäänkirjautuminen on toteutettu generoitujen avainparien avulla.

Tässä työssä pohjana käytetään Linux-pohjaista Ubuntu 11.10 -käyttöjärjestelmää, joka on luotu yhteisön kautta saatavilla olevasta levykuvasta. Tarvittavat palvelinohjelmistot määräytyvät pääosin tavoitesovelluksen, tässä tapauksessa Drupal-järjestelmän järjestelmävaatimuksista[19]. Drupal 7:n ilmoitetut järjestelmävaatimukset ovat:

- 15 megatavua levytilaa
- Apache 1.3, Apache 2.x tai Microsoft IIS -HTTP-palvelinohjelma
- MySQL 5.0.15 tai uudempi PDO-tuella, PostgreSQL 8.3 tai uudempi PDO-tuella tai SQLite 3.3.7 tai uudempi
- PHP 5.2.5 tai uudempi, versio 5.3 suositeltava

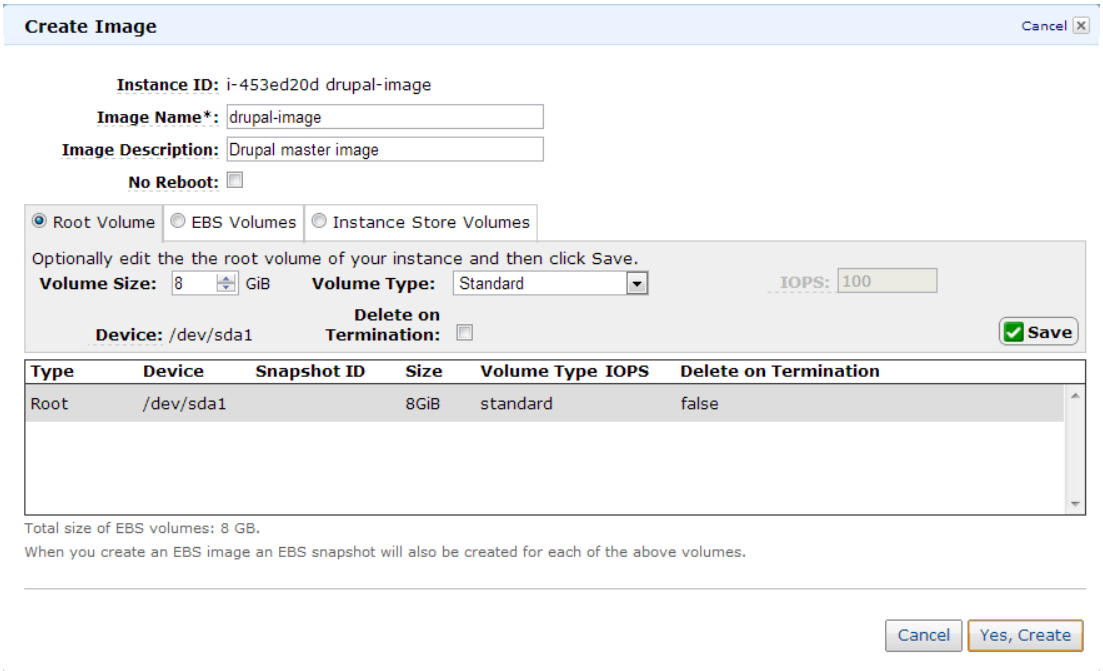
Käytettäväksi HTTP-palvelimeksi valitaan Apache 2.x ja tietokantapalvelimeksi MySQL. Tässä työssä omalle Linux-palvelimelle asennetaan vain Apache 2.x- ja PHP-ohjelmistot. Tietokantana käytettävä MySQL-palvelin luodaan Amazonin omilla työkaluilla ja siirretään sen ylläpitovastuu Amazonille.

[19] `http://www.drupal.org/requirements/`

5.1.1 Levykuvan luominen

Levykuvalla tarkoitetaan tiedostoa, joka on täydellinen kopio järjestelmän levystä. Se sisältää kaikki kyseisen partition tiedot, mukaan lukien käynnistyssektorit. Levykuvaa käytetään, kun halutaan palauttaa järjestelmä siihen tilaan kuin missä se oli levykuvaa ottaessa. (Webopedia, 2012).

Levykuvan luominen on välttämätön välivaihe, jotta skaalautuvuus saadaan toteutettua. Koko järjestelmän pohjana toimii yksi EC2-instanssi, josta levykuvan avulla voidaan monistaa uusia palvelininstansseja. Kuvassa 5.1 on esitetty, miten levykuvan luonti Amazonin työkalun avulla tapahtuu.



Kuva 5.1: Levykuvan luominen EC2-instanssista

Jokaiselle levykuvalle luodaan oma yksilöivä tunniste levykuvan luonnin yhteydessä. Tätä tunnistetta tarvitaan, kun halutaan hyödyntää automaattista skaalautuvuutta. Järjestelmälle on kerrottava, minkä levykuvan pohjalta uusia palvelininstansseja luodaan. Luvussa 5.5 on esitetty, miten tämä tehdään ohjelmallisesti.

5.1.2 Kuormantasaaja

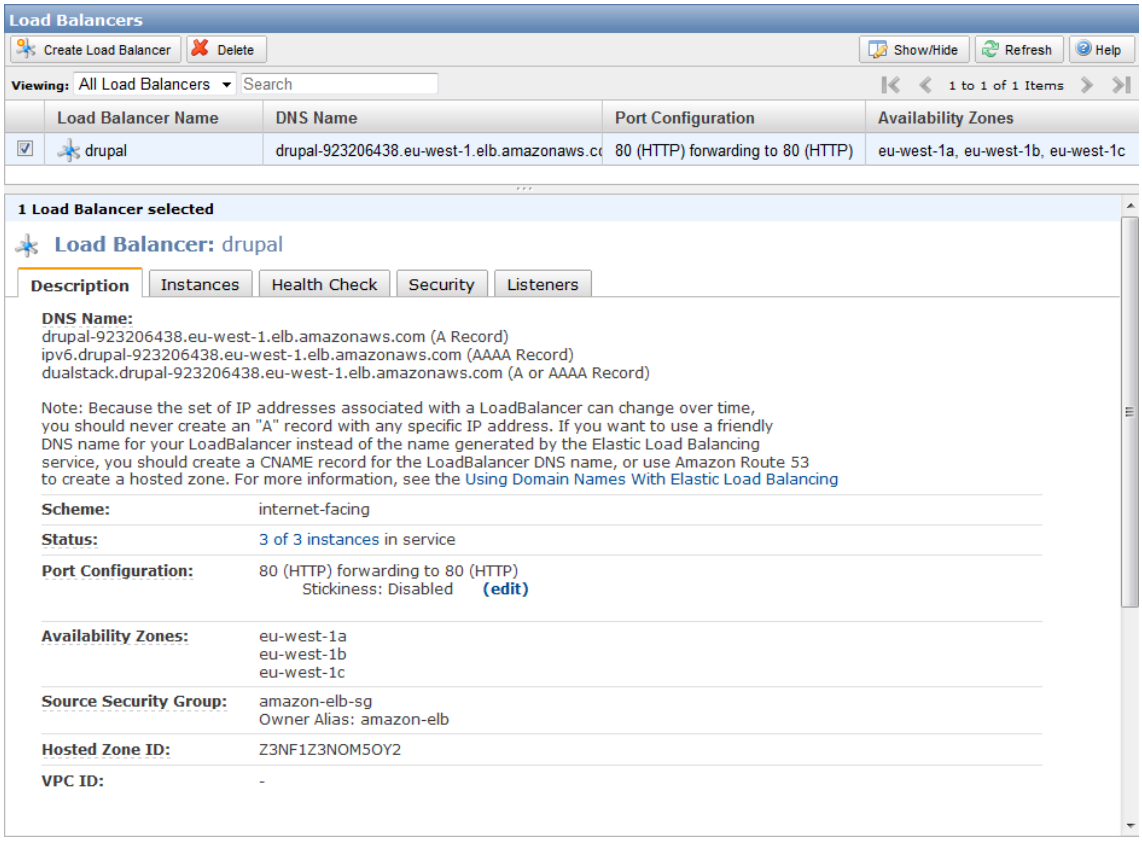
Kuormantasaajan tarkoitus on jakaa sisään tuleva liikenne tasaisesti kaikille käytössä oleville EC2-instansseille. Sen avulla voidaan saavuttaa entistä parempi järjestelmän vikasietoisuus. Amazon ELB (engl. Elastic Load Balancing) monitoroi EC2-instansseja ja havaitessaan rikkinäisen EC2-instanssin, se automaattisesti poistaa kyseisen instanssin käytettävissä olevien resurssien listalta. Tämän jälkeen mikään uusi kysely enää ohjaudu rikkinäiselle palvelimelle. Kun palvelin on taas

käytössä, niin se lisätään taas automaattisesti käytettävissä oleviin resursseihin. (Amazon ELB, 2012).

Uuden kuormantasaajan luonnin yhteydessä kuormantasaajalle määritetään nimi, uudelleenohjattavat portit sekä käytettävissä olevat EC2-instanssit. Lisäksi määritellään, miten kuormantasaaja tunnistaa ehjän tai rikkinäisen instanssin. Tämä voi olla esimerkiksi yksinkertainen HTTP-kysely. Jokaiselle kuormantasaajalle luodaan automaattisesti oma, yksilöllinen DNS-nimi. Tämä DNS-nimi on kuitenkin usein niin monimutkainen, ettei sitä voi suoraan käyttää verkkopalvelun osoitteena. Tähän on ratkaisuna käyttää nimipalvelimen CNAME-kenttää ja luoda oma osoite. Tässä työssä Drupal-sivustolle on luotu seuraavanlainen oma CNAME-kenttä:

```
$ host drupal.cloud.samuli.info
drupal.cloud.samuli.info is an alias for drupal-923205.eu-west-1.elb.amazonaws.com.
drupal-923205.eu-west-1.elb.amazonaws.com has address 176.34.113.124
drupal-923205.eu-west-1.elb.amazonaws.com has address 46.137.186.213
drupal-923205.eu-west-1.elb.amazonaws.com has address 54.247.121.170
```

Drupal-sivuston tapauksessa kuormantasaaja on asetettu uudelleenohjaamaan jokainen HTTP- ja HTTPS-kysely käytettävissä oleville EC2-instansseille. Nyt osoitteessa http://drupal.cloud.samuli.info/ ei vastaa enää omassa hallinnassa oleva EC2-instanssi, vaan Amazonin hallinnoima kuormantasaajapalvelin. Kuormantasaaja jakaa jokaisen kyselyn tasaisesti eri palvelimille, jolloin esimerkiksi yksi sivulataus voi koostua useiden eri palvelinten prosessoimasta datasta. Kuvassa 5.2 on esitetty luotu kuormantasaaja.



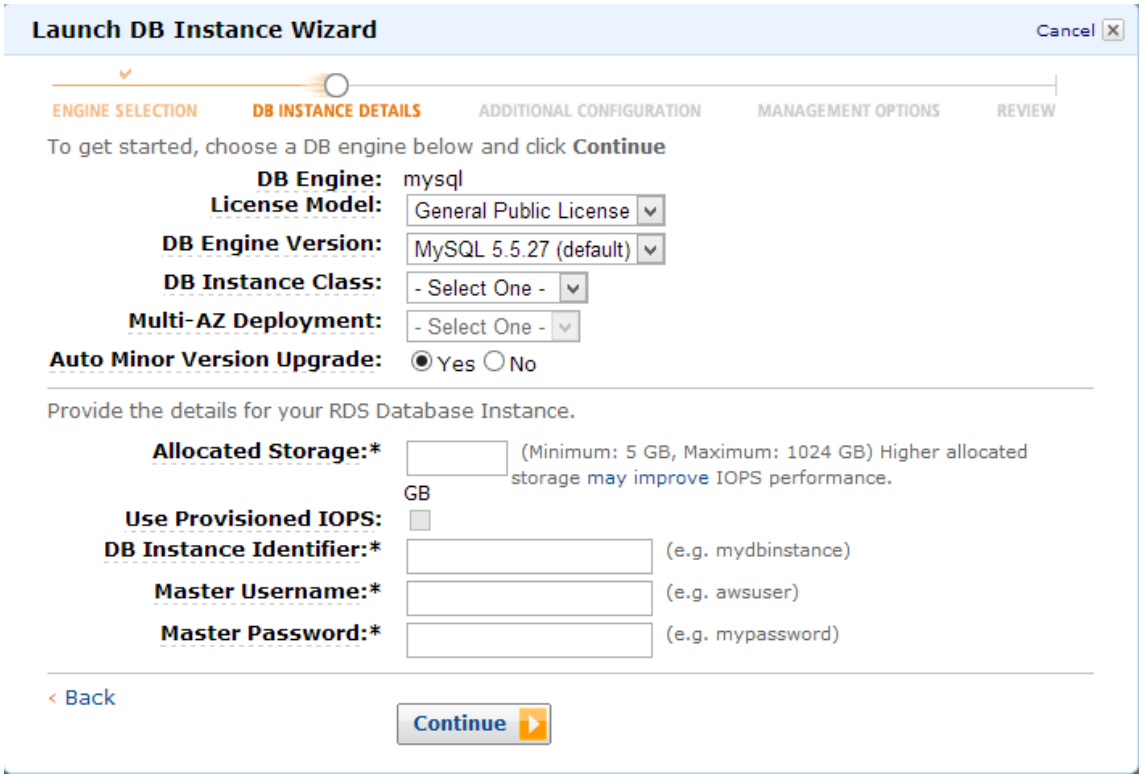
Kuva 5.2: Kuormantasaaja

5.2 RDS-tietokantapalvelin

Amazon RDS (engl. Relational Database Service) tarjoaa mahdollisuuden luoda uusia palvelininstansseja, jotka on rajattu vain tietokantakäyttöön ja niiden ylläpito on siirretty Amazonille. Käyttäjällä on hallintakonsolin kautta muun muassa mahdollisuus luoda ja poistaa omia tietokantainstansseja, luoda vain luku -replikoita tietokantainstansseista sekä muokata tietokantapalvelininstanssin käytettävissä olevia resursseja, eli esimerkiksi levytilaa ja laskentatehoa.

5.2.1 Tietokantapalvelimen luominen

Uuden tietokantapalvelimen luomisen yhteydessä on mahdollisuus valita käytettävä tietokantaohjelmisto sekä tietokantaohjelmiston tietty versio. Tässä työssä käytettäväksi tietokantaohjelmistoksi on valittu MySQL. Kuva 5.3 havainnollistaa uuden tietokantapalvelimen luomisessa tarvittavia parametreja. Jokaiselle luotavalle tietokantapalvelimelle on annettava lisäksi yksilöivä tunniste. Lisäksi luomisvaiheessa on ilmoitettava, haluaako käyttää Multi-AZ-toiminnallisuutta. Sen toimintaperiaatteet on kuvattu luvussa 4.1.2.



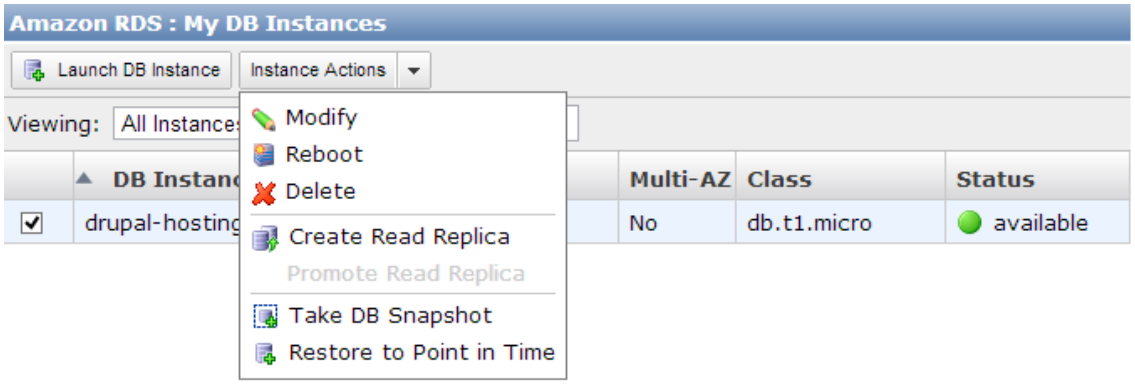
Kuva 5.3: RDS-tietokantapalvelimen luominen

5.2.2 Vain luku -replikat

Replikointi tarkoittaa tietokannan sisällön kopioimista ensisijaiselta tietokantapalvelimelta yhdelle tai useammalle toissijaiselle tietokantapalvelimelle. Replikointi tapahtuu asynkronisesti, eli toissijaisen tietokantapalvelimen ei tarvitse olla koko ajan yhteydessä ensisijaiseen tietokantapalvelimeen. (MySQL Replication, 2012).

Tietokantakyselyt voidaan jakaa kahteen luokkaan: kirjoitus- ja lukukyselyihin. Lukukyselyt eivät muuta tietokannan tilaa, eli ne voidaan suorittaa erillisessä vain luku -tilassa. Skaalautuvuuden näkökulmasta tätä ominaisuutta voidaan hyödyntää luomalla kirjoitus- ja lukukyselyille erilliset tietokantapalvelimet. Kaikki tietokantaohjelmistot eivät tue tätä toiminnallisuutta, mutta tässä työssä käytettävä MySQL mahdollistaa replikoinnin.

Amazonin hallintapaneelin avulla vain luku -replikoiden luonti on helppoa. Valitaan jokin master-tietokanta, jolloin hallintapaneelista löytyy kuvan 5.4 mukaisesti “Create read replica”-painike. Vain luku -replikoita on mahdollista luoda rajaton määrä.

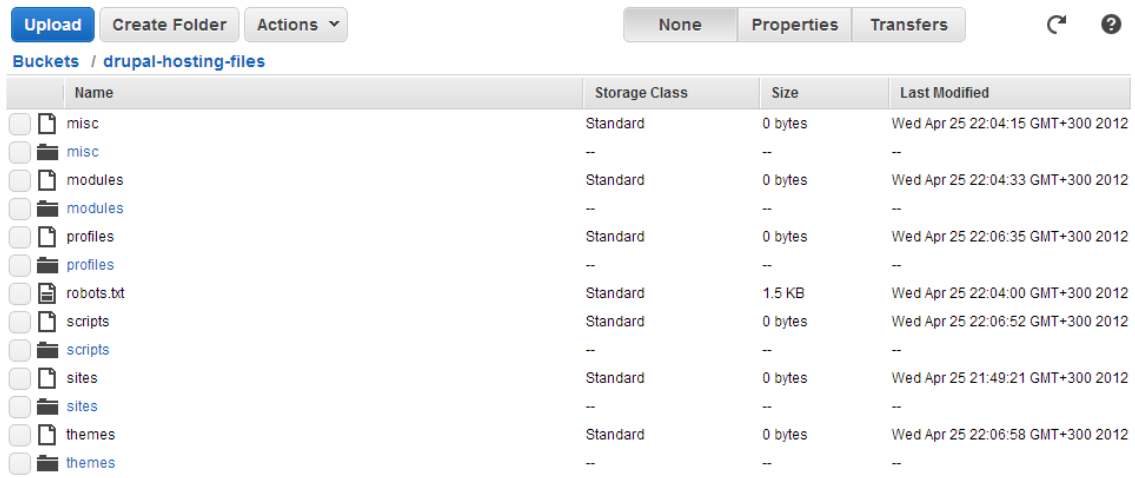


Kuva 5.4: Vain luku -replikan luonti

5.3 S3-levyjärjestelmä

Käyttäjillä on mahdollisuus luoda rajaton määrä S3-säiliöitä. Säiliön luonnissa on syytä kiinnittää huomiota siihen, mihin palvelinkeskukseen se luodaan. Palvelimen ja käytetyn S3-levyjärjestelmän olisi hyvä sijaita samassa palvelinkeskuksessa. Näin varmistetaan nopea verkkoyhteys palvelimelta levylle.

Tässä esimerkissä luodaan “drupal-hosting-files”-niminen S3-säiliö. Säiliötä käytetään Drupalin staattisten tiedostojen säilyttämiseen ja jakamiseen loppukäyttäjälle sisällönjakeluverkon kautta. Kuvassa 5.5 on esitetty luotu S3-säiliö sekä sinne tallennetut Drupalin staattiset tiedostot. Luvussa 5.4 on esitetty tarkemmin, miten S3-säiliöön tallennettua tietoa voidaan jakaa käyttäen sisällönjakeluverkkoa.



Kuva 5.5: S3-säiliön sisältö ja sen hallinta

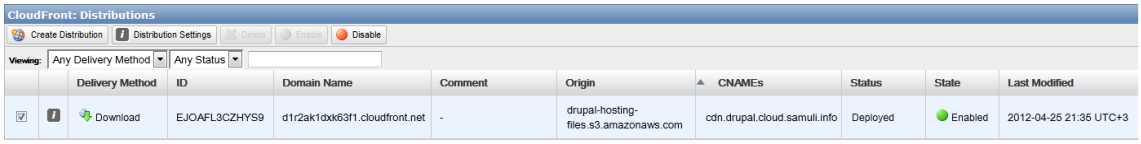
Levyjärjestelmän soveltuvuus tavallisten sovellusten käyttöön olisi kuitenkin huono, mikäli sen käyttö vaatisi verkkoselainta. Tämän vuoksi S3-levyjärjestelmästä löytyykin monipuoliset REST- ja SOAP-rajapinnat, joiden avulla levyjärjestelmän käyttö onnistuu suoraan sovelluksestakin. Tähän tarkoitukseen on myös olemassa

s3fs-työkalu, jonka avulla on mahdollista liittää S3-säiliö Linux-käyttöjärjestelmän johonkin liitospisteeseen. Tästä työkalusta ja sen käytöstä on kerrottu lisää luvussa 6.2.

5.4 CloudFront-sisällönjakeluverkko

Amazon CloudFront toimii rajapintana Amazon S3 -palveluun. Palvelun käyttöönoton yhteydessä määritetään, mihin Amazon S3 -säiliöön osoite liitetään. Tämän jälkeen palvelu luo sille uniikin URL-osoitteen. Osoite on myös mahdollista personoida erillisen CNAME-kentän avulla. Kuvassa 5.6 on esitetty, miten "drupal-hosting-files"-niminen S3-säiliö on liitetty CloudFront-palveluun, ja sille on lisätty CNAME-kentäksi "cdn.drupal.cloud.samuli.info". CNAME-kentän käyttöönotto vaatii lisäksi kyseisen kentän lisäämistä CNAME-kentästä vastaavalle nimipalvelimelle. Nyt nimipalvelinkyselyn tuloksena palautuvat loppukäyttäjän sijainnista riippuen Amazonin sisällönjakeluverkon lähimpien reunapalvelinten IP-osoitteet:

```
$ host cdn.drupal.cloud.samuli.info
cdn.drupal.cloud.samuli.info is an alias for d1r2ak1dxk63f1.cloudfront.net.
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.51
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.199
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.135
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.229
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.21
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.139
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.107
d1r2ak1dxk63f1.cloudfront.net has address 205.251.219.240
```



Kuva 5.6: S3-säiliön liittäminen CloudFront-palveluun

Sisällönjakeluverkon käyttöönotto vaatii siis aina erillisen osoitteen käyttöä. Verkkopalvelun tapauksessa sovelluksen on itse tunnistettava, onko sisältö staattista vai dynaamista. Staattinen sisältö jaetaan käyttäen sisällönjakeluverkkoa ja dynaaminen suoraan palvelimelta. Esimerkiksi Drupal-sivuston tapauksessa Drupalin ydin itse ei tue tätä toiminnallisuutta. Sisällönjakeluverkon hyödyntäminen vaatii siis erillisen moduulin asentamista. Tästä ominaisuudesta on kerrottu tarkemmin luvussa 6.3.

5.5 Automaattinen skaalautuvuus

Pilvipalveluiden kustannustehokkuus syntyy mahdollisuudesta käyttää vain kyseisellä hetkellä tarvittu määrä resursseja. Tiettynä vuorokaudenaikana palvelulla saat-

taa olla kymmenkertainen käyttäjämäärä toiseen vuorokaudenaikaan verrattuna. Varatut resurssit maksavat, joten ylimääräiset resurssit on kustannustehokkuuden vuoksi syytä poistaa käytöstä, kun niille ei ole tarvetta. Yksittäisiin kävijäpiikkeihin on kuitenkin myös syytä varautua, eli käytettävissä olevia resursseja pitää olla mahdollista kasvattaa vuorokaudenajasta riippumatta.

Amazon CloudWatch tarjoaa mahdollisuuden mitata EC2-resurssien käyttöarvoja. Tällaisia käyttöarvoja voivat olla muun muassa prosessorin käyttöaste, verkkoliikenteen määrä tai levyn käyttö. Arvoja voidaan mitata instanssikohtaisesti tai kaikkien instanssien keskiarvona. Näiden mittausten perusteella on mahdollista luoda hälytyksiä. Hälytys voi syntyä esimerkiksi, kun kaikkien instanssien yhteinen prosessorin käyttöaste on noussut yli 50 %:n. Erilaisille hälytyksille voidaan määrittää toimenpiteitä. Esimerkiksi prosessorin käyttöasteen noustessa yli 50 %:n, voidaan suorittaa toimenpide, joka luo pilveen kaksi uutta EC2-instanssia. Jos tämäkään ei vielä auta ja syntyy uusi hälytys, niin pilveen luodaan määritettyyn rajaan asti uusia instansseja niin kauan, ettei hälytyksiä enää tule. Vastaavasti voidaan luoda alhaisen prosessorikäytön hälytys, joka tarkoittaa, että pilvestä tulee poistaa instansseja. Poistaminen voidaan kuitenkin tehdä hillitymmin ja poistaa aina vain yksi instanssi kerrallaan.

Hälytysten ja toimenpiteiden luonti onnistuu myös AWS:n hallintakonsolin avulla, mutta siihen on tarjolla myös helppokäyttöisiä komentorivipohjaisia työkaluja. Komentorivipohjaiset työkalut toimivat Amazonin tarjoaman API:n avulla. Niiden käyttäminen vaatii omien salaisten avaimien luomista, jonka jälkeen API:n käyttö on mahdollista miltä palvelimelta tahansa. Seuraavaksi esitellään ylläolevan esimerkin mukaiset toimenpiteet, joiden avulla automaattinen skaalautuvuus saadaan toteutettua. Komennot on suoritettu Linux-pohjaisessa käyttöjärjestelmässä. Käytetyistä työkaluista löytyy lisää tietoa luvusta 4.1.

Ensimmäinen vaihe on käynnistyskonfiguraation luominen. Käynnistyskonfiguraatiossa kerrotaan muun muassa, että mistä levykuvasta tarpeen vaatiessa halutaan luoda uusi instanssi, miten paljon instanssilla on resursseja käytössään sekä mihin maantieteelliseen sijaintiin palvelin luodaan:

```
$ ./as-create-launch-config drupal --image-id ami-dde4dfa9 --instance-type \
m1.small --key drupal-sites --group all --region eu-west-1
OK-Created launch config
```

Nyt käynnistyskonfiguraatio on luotu. Seuraavaksi luodaan oma ryhmänsä automaattiselle skaalaukselle, joka käyttää aikaisemmin luotua käynnistyskonfiguraatiota. Parametreina kerrotaan muun muassa, mihin maantieteelliseen sijaintiin palvelin luodaan, mitä saatavuusalueita halutaan käyttää, mitkä ovat palvelininstanssien minimi- ja maksimimäärät sekä mihin kuormantasaajaan uudet instanssit halutaan liittää:

```
$ ./as-create-auto-scaling-group drupal-group --launch-configuration drupal \
```

```
--availability-zones eu-west-1b eu-west-1c --min-size 2 --max-size 10 \
--load-balancers drupal --region eu-west-1
OK-Created AutoScalingGroup
```

Tämän jälkeen tarvitaan vielä säännöt, että milloin ja miten uusia palvelininstansseja halutaan luoda ja poistaa. Parametreina kerrotaan muun muassa, että mihin automaattisen skaalauksen ryhmään sääntö kuuluu, mitä hälytyksen tullessa tehdään, eli montako uutta instanssia luodaan tai montako vanhaa instanssia poistetaan sekä jälleen, että missä maantieteellisessä sijainnissa tämä halutaan tehdä. Seuraavassa luodaan sekä uuden instanssin luontiin että vanhan instanssin poistoon liittyvät säännöt:

```
$ ./as-put-scaling-policy drupal-scale-up --auto-scaling-group drupal-group \
--adjustment=2 --type ChangeInCapacity --cooldown 300 --region eu-west-1
arn:aws:autoscaling:eu-west-1:215462922696:scalingPolicy:237dd9c9-6348-49c2-a905
-720111c0f44f:autoScalingGroupName/drupal-group:policyName/drupal-scale-up

$ ./as-put-scaling-policy drupal-scale-down --auto-scaling-group drupal-group \
--adjustment=-1 --type ChangeInCapacity --cooldown 300 --region eu-west-1
arn:aws:autoscaling:eu-west-1:215462922696:scalingPolicy:04e90367-4496-4aed-9092
-df5e7e1d7104:autoScalingGroupName/drupal-group:policyName/drupal-scale-down
```

Komennot antavat vastauksena yksilöivän nimen, jonka avulla hälytykselle voidaan liittää toimenpide. Seuraavaksi luodaan sekä korkean, että matalan prosessorikäytön hälytykset. Korkean prosessikäytön hälytys syntyy, kun kaikkien instanssien yhteinen prosessorien käyttöaste on yli 50 % ja alhaisen, kun käyttöaste on alle 15 %. Parametrina hälytyksen luonnissa annetaan muun muassa aikaisemman komennon palauttama yksilöivä tunniste sekä vertailutyyppi- ja arvo:

```
$ ./mon-put-metric-alarm high-cpu --comparison-operator GreaterThanThreshold \
--evaluation-periods 1 --metric-name CPUUtilization --namespace "AWS/EC2" \
--period 600 --statistic Average --threshold 50 --region eu-west-1 \
--alarm-actions arn:aws:autoscaling:eu-west-1:215462922696:scalingPolicy: \
92f212f2-fc0c-42c8-bc5a-7cc2d451b5e5:autoScalingGroupName/drupal-group: \
policyName/drupal-scale-up --dimensions "AutoScalingGroupName=drupal-group"
OK-Created Alarm

$ ./mon-put-metric-alarm low-cpu --comparison-operator LessThanThreshold \
--evaluation-periods 1 --metric-name CPUUtilization --namespace "AWS/EC2" \
--period 600 --statistic Average --threshold 15 --region eu-west-1 \
--alarm-actions arn:aws:autoscaling:eu-west-1:215462922696:scalingPolicy: \
88b75aed-11a6-4ca7-abb7-2ce67c66100d:autoScalingGroupName/drupal-group: \
policyName/drupal-scale-down --dimensions "AutoScalingGroupName=drupal-group"
OK-Created Alarm
```

Nyt järjestelmään on luotu kaksi erilaista hälytystä sekä näille vastaavat toimenpiteet. Järjestelmästä pitäisi siis löytyä aina vähintään kaksi käynnissä olevaa instanssia, mutta maksimissaan kymmenen. Jos instanssien yhteinen prosessorin käyttöaste nousee yli 50 %, järjestelmään luodaan kaksi uutta instanssia ja vastaavasti yksi instanssi poistetaan, kun instanssien yhteinen prosessorin käyttöaste on alle 15 %.

6 Drupalin käyttöönotto

Luvussa 5 luotiin skaalautuva ja vikasietoinen pilvipalvelinympäristö Amazonin tarjoamien työkalujen avulla. Tässä luvussa hyödynnetään luotua palvelinalustaa ja asennetaan siihen esimerkkisovellus. Esimerkkisovelluksena käytetään Drupal-sisällönhallintajärjestelmää.

Pilvipalvelinalustan luonnissa on kiinnitetty erityistä huomiota alustan skaalautuvuuteen. Näin ollen myös Drupalin asennuksessa ja konfiguroinnissa on kiinnitettävä siihen huomiota. Työn tavoitteena on saada Drupal käyttämään mahdollisimman hyvin hyödyksi kaikkia pilvipalvelinympäristön tarjoamia resursseja.

6.1 Drupalin esittely

Drupal[20] on erittäin modulaarinen avoimen lähdekoodin sisällönhallintajärjestelmä ja sisällönhallintakehys. Ohjelmiston kehityksessä on panostettu huomattavasti koodin laajennettavuuteen, standardien noudattamiseen sekä koodin selkeyteen. Ytimen koodi on yhtenäistä ja se noudattaa tiukkoja PEAR-ohjelmointistandardeja, jotka on sovelletusti määritelty tarkemmin Drupalin dokumentaatiossa[21]. (VanDyk & Westgate, 2007, s. 1).

Drupalin uusin vakaa versio 7 julkaistiin vuoden 2011 alussa[22]. Ytimen kehitystyö on kuitenkin aktiivista ja uusia ytimen versioita julkaistaan useita kertoja vuodessa. Kirjoitushetkellä uusin Drupal 7:n versio on 7.17. Drupalin ytimen kehityksestä vastaa Drupal-käyttäjien yhteisö. Yhteisö on hyvin laaja ja järjestäytynyt. Tämä takaakin sen, että mahdollisten tietoturva-aukkojen korjaaminen tapahtuu nopeasti.

Drupalin ytimien 6 ja 7 välillä on isoja eroja. Kuten aikaisemmin jo todettiin, koostuu Drupalin ydin useammasta moduulista. Ytimen rakenne on kuitenkin oletuksena hyvin suppea, eli tarjoaa vain sisällönhallintajärjestelmän välttämättömän pohjan. Useiden eri toiminnallisuuksien lisääminen verkkopalveluun vaatii kolmannen osapuolen moduuleita. Drupal 7:n ytimeen on nyt sisäänrakennettu useita Drupal 6:n käytössä hyväksi ja lähes pakollisiksi havaittuja moduuleita. On syytä huomata, että vanhemman version moduulit eivät ole yhteensopivia uudemman version kanssa. Drupal 6 -sivustolla käytössä olleita moduuleita ei siis ole välttämättä mahdollista käyttää Drupal 7 -sivustolla ollenkaan. Moduulin kehittäjän pitää luoda moduulista uusi versio, joka on toteutettu Drupal 7:n rajapintoja käyttäen.

Drupal 7:n myötä koko käyttöliittymä on kokenut suuren muutoksen. Käyt-

[20] `http://www.drupal.org/`
[21] `http://drupal.org/documentation`
[22] `http://drupal.org/drupal-7-released`

töliittymän toteutuksessa on hyödynnetty Ajax-tekniikkaa. Garrettin (2005, ss. 1-2) mukaan Ajax ei ole teknologia. Se on kokoelma useista teknologioista, jotka nivoutuvat tehokkaasti yhteen aivan uudella tavalla. Ajaxin avulla Drupal-sivusto pystyy suorittamaan operaatioita ja noutamaan tietoa ilman, että koko näytettävää sivua on pakko ladata uudestaan. Tavallisen verkkosivun lataamisen sijaan session aluksi selain lataakin Ajax-toteutuksen, joka vastaa sivun käyttöliittymän renderöinnistä sekä palvelimen kanssa kommunikoinnista. Tällöin sivuston on siis mahdollista toimia niin, ettei kokonaista uutta sivulatausta tarvitse enää tehdä.

6.2 Instanssien väliset jaetut resurssit

Skaalautuvuus ja usean palvelimen rinnakkainen käyttö aiheuttavat uusia haasteita sovelluksen toiminnalle. Drupal käyttää pääasiassa tietokantaa tiedon tallettamiseen, mutta myös levyjärjestelmään talletetaan paljon tietoa. Kaikki tietokannan kirjoituskyselyt kohdistuvat aina master-tietokantapalvelimeen, joka replikoi tiedot edelleen vain luku -palvelimille. Tietokantapalvelimet pysyvät siis automaattisesti tällä arkkitehtuurimallilla ajan tasalla. Talletettujen staattisten tiedostojen suhteen tilanne ei ole aivan yhtä yksinkertainen. Yksi ratkaisuvaihtoehto tähän voisi olla käyttää yhtä erillistä tiedostopalvelinta, joka liitettäisiin hakemistoksi jokaiseen instanssiin. Tämä kuitenkin aiheuttaisi paljon kuormaa tälle yhdelle tiedostopalvelimelle ja tämän lisäksi tiedostopalvelimen vikaantuessa koko palvelu olisi staattisten tiedostojen osalta saavuttamattomissa.

Amazon S3 tarjoaa tähän rajapintojensa avulla helpon ratkaisun. Palveluun on mahdollista luoda rajaton määrä säiliöitä, eli esimerkiksi yksi säiliö jokaista Drupal-sivustoa kohti. Linux-käyttöjärjestelmissä on mahdollista hyödyntää s3fs[23]-työkalua, joka liittää S3-säiliön määritettyyn hakemistopolkuun. Työkalun avulla “s3-drupal”-nimisen S3-säiliön liittäminen hakemistoon “/var/www/drupal/sites/drupal.domain.com/files” tapahtuu komennolla:

```
$ s3fs -o passwd_file=/etc/passwd-s3fs s3-drupal:/ \
/var/www/drupal/sites/drupal.domain.com/files
```

Komento tarvitsee lisäksi parametrinaan joko suoraan syötetyt AWS:n avaintiedot tai avaintiedoston sijainnin. API:n käytössä tarvituista avaimista on kerrottu tarkemmin luvussa 4.1.1. Suorituskyvyn parantamiseksi S3-säiliön sisältöä on myös mahdollista asettaa välimuistiin. Välimuistiin asetettu tiedosto löytyy suoraan paikalliselta levyltä, eikä tällöin muodostu viivettä, joka muuten syntyy verkon yli haettavasta tiedosta. Tämä onnistuu parametrilla “use_cache=/polku/välimuistihakemistoon”.

[23] `http://code.google.com/p/s3fs/wiki/FuseOverAmazon`

6.3 Sisällönjakeluverkon hyödyntäminen

Amazon CloudFront tukee staattisen, dynaamisen ja suoratoistodatan jakamista loppukäyttäjälle. Drupal-sivuston tapauksessa sisältö on joko staattista tai dynaamista. Dynaamisen sisällön luonti vaatii kuitenkin aina paljon sellaista informaatiota, jota sisällönjakeluverkon palvelimella ei ole käytössään. Näin ollen sisällönjakeluverkkoa on Drupal-sivuston tapauksessa perusteltua käyttää vain staattisen sisällön jakamiseen.

6.3.1 Sisällön eri tyypit

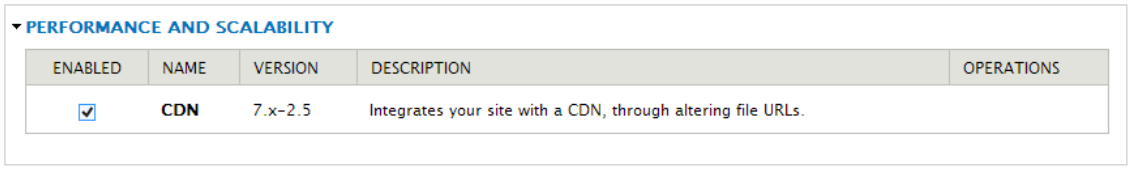
Yksittäiset verkkosivut voivat olla luonteeltaan joko staattisia tai dynaamisia. Staattisten sivujen sisältö on vakio. Dynaamisten sivujen sisältö puolestaan luodaan jokaisella sivulatauksella uudestaan. Sen sisältö saattaa riippua useistakin eri tekijöistä, kuten esimerkiksi käyttäjän lomakkeelle syöttämistä arvoista tai käyttäjän käyttöoikeuksista. (Vrt. Ricca & Tonella, 2001, s. 26). Sisällönjakeluverkot soveltuvat Drupal-sivuston tapauksessa vain staattisen sisällön jakamiseen. Dynaamisten sivujen luominen vaatii aina sellaista palvelimen sisäistä tietoa, jota sisällönjakeluverkon palvelimella ei ole käytettävissään, jotta se voisi luoda tämän sivun.

Suorituskykyä voidaan kuitenkin parantaa asettamalla dynaamisesti luotuja sivuja välimuistiin. Tällöin nämä välimuistiin asetetut tiedostot voidaan jakaa käyttäjille sisällönjakeluverkon kautta, jolloin palvelimen rasitusta saadaan pienennettyä. Drupal tarjoaa työkalun, jonka avulla voidaan hallita välimuistin sisältöä. Työkalun avulla voidaan asettaa välimuistiin asetettujen tiedostojen minimi- ja maksimielinaika. Minimielinaika varmistaa, ettei Drupal luo sivua uudestaan ennen kuin kyseinen aika on kulunut. Maksimielinaika puolestaan takaa sen, että viimeistään tämän ajan kuluttua Drupal uudistaa sivun sisällön ennen kuin sisältö toimitetaan loppukäyttäjälle.

6.3.2 CDN-moduuli

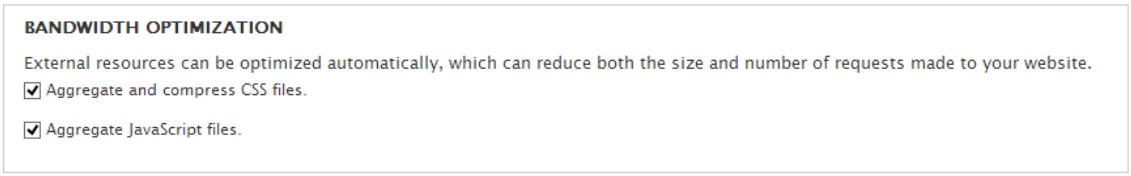
Sisällönjakeluverkon hyödyntämiseksi Drupaliin on saatavilla kolmannen osapuolen CDN-moduuli[24]. Moduulin avulla on mahdollista manipuloida staattisten tiedostojen URL-osoitetta niin, että polku säilytetään, mutta lähdepalvelimen osoite vaihdetaan. Tällöin verkkoselain noutaa staattiset tiedostot erilliseltä sisällönjakeluverkon palvelimelta. Moduulin asentaminen tapahtuu samalla tavalla kuin minkä tahansa muunkin. Sillä ei ole riippuvuussuhteita muihin moduuleihin. Asennuksen jälkeen moduuli aktivoidaan Drupalin moduulien hallinnasta kuvan 6.1 osoittamalla tavalla.

[24] `http://drupal.org/project/cdn`



Kuva 6.1: CDN-moduulin aktivointi

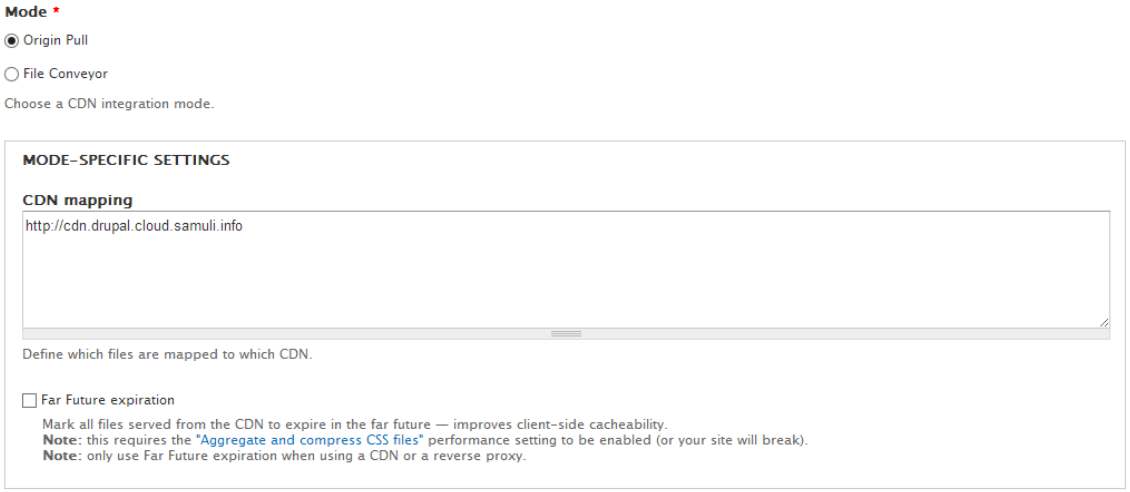
CDN-moduuli vaatii toimiakseen tiettyjen Drupalin ominaisuuksien kytkemistä päälle. CSS- ja JavaScript-tiedostojen pakkaaminen on otettava käyttöön kuvan 6.2 osoittamalla tavalla. Pakkaamisessa useita CSS- tai JavaScript-tiedostoja yhdistetään yhdeksi tiedostoksi. Lisäksi tiedoston rakennetta muokataan niin, että se menee mahdollisimman pieneen tilaan. Isolla Drupal-sivustolla voi olla käytössään useita kymmeniä moduuleita. Jokaisella näistä moduulista voi olla yksi tai useampi CSS-tiedosto, jotka se lisää järjestelmään. CSS-tiedostojen määrän myötä voi myös syntyä yllättäviä ongelmia. Esimerkiksi Microsoft Internet Explorer -verkkoselain lukee vain 32 ensimmäistä CSS-tiedostoa[25]. Tällöin on mahdollista, että useita kriittisiä tyyliasetuksia jää lukematta ja sivuston ulkoasu ei toimi oikein. CSS-tiedostojen pakkaaminen siis ratkaisee myös samalla tämän ongelman.



Kuva 6.2: CSS- ja JavaScript-tiedostojen pakkaaminen

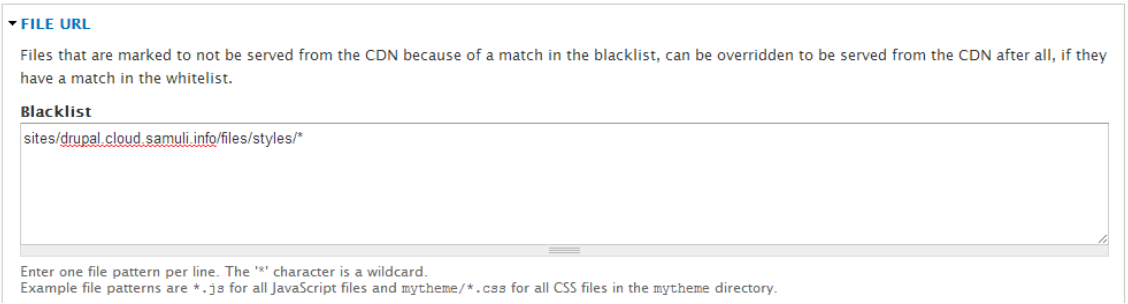
Ennen CDN-moduulin varsinaista käyttöönottoa on kopioitava Drupalin staattiset tiedostot S3-levyjärjestelmään. Tämän lisäksi on syytä varmistaa, että ne löytyvät sisällönjakeluverkon URL-osoitteesta. S3-levyjärjestelmä voi vaatia erillisten käyttöoikeuksien määrittämisen, että tiedostoihin on lukuoikeus kaikilla käyttäjillä. Tässä esimerkissä käytetään “Origin Pull”-tilaa, eli tiedostot haetaan suoraan Amazonin sisällönjakeluverkosta. Kuvassa 6.3 on esitetty, miten CDN-moduulille kerrotaan URL-osoite, josta sisällönjakeluverkon palvelin ja näin ollen Drupalin staattiset tiedostot löytyvät. Tämän jälkeen moduuli osaa itse muuntaa staattisten tiedostojen URL-osoitteen niin, että se osoittaa sisällönjakeluverkon palvelimelle.

[25] `http://support.microsoft.com/kb/262161`



Kuva 6.3: URL-osoitteiden uudelleenkirjoitus CDN-moduulin avulla

CDN-moduuli toimii lähes kaikissa tilanteissa asennuksen jälkeen oikein. On kuitenkin erikoistilanteita, jolloin staattinenkin sisältö halutaan välittää loppukäyttäjälle alkuperäisestä sijainnista. Lisäksi Drupalin ytimestä löytyvän ImageCache-moduulin käsittelemät tiedostot saattavat näyttää staattisilta, vaikka ne eivät olekaan. URL-osoite ohjaa siis näennäisesti kuvatiedostoon, mutta ennen kuvan esittämistä ImageCache-moduuli prosessoi sen. Prosessointi saattaa olla esimerkiksi kuvan pienentäminen tiettyyn maksimikokoon. Näissä tapauksissa tiedostoa ei voida etsiä suoraan sisällönjakeluverkon palvelimelta, koska varsinaista kuvatiedostoa ei ole siellä. CDN-moduulista löytyy mahdollisuus merkitä erilaisia osoitteita niin sanotulle mustalle listalle, joiden URL-osoitetta ei manipuloida osoittamaan sisällönjakeluverkon palvelimelle. Kuvassa 6.4 on esitetty, miten ImageCache-moduulin kuvien noutoa sisällönjakeluverkosta ei sallita.



Kuva 6.4: CDN-moduulin musta lista

7 Ominaisuuksien arviointi

Työssä toteutettua pilvipalvelinympäristön luomista sekä Drupal-järjestelmän asentamista siihen arvioidaan kahdesta eri näkökulmasta. Ensinnäkin verrataan, miten pilvipalvelinympäristö ja tavallinen palvelin eroavat toisistaan ominaisuuksiensa ja rajoitteidensa suhteen. Lisäksi verrataan Drupal-järjestelmän soveltuvuutta pilvipalvelinympäristöön tavalliseen yhden palvelimen järjestelmään verrattuna.

7.1 Arviointimenetelmät

Monikriteerimenetelmät (engl. multi-criteria decision analysis (MCDA) tai multi-criteria decision making (MCDM)) tarjoavat systemaattisen lähestymistavan yhdistää monen tasoista tietoa päätöksenteon tukemiseksi. MCDA-mallien avulla pystytään yhtäaikaisesti keskenään vertailemaan ominaisuuksia ja teknologioita, joilla ominaisuudet on toteutettu. MCDA-mallien avulla on mahdollista arvioida tuloksia, kun arvioitavana olevat ominaisuudet ovat ristiriidassa keskenään. MCDA-malli osoittaa vaihtoehtojen väliset ristiriidat ja luo pohjan tulosten perusteella läpinäkyvästi tehtävälle päätökselle. MCDA-malli ei pyri esittämään yksikäsitteistä ratkaisua ongelmaan, vaan sen avulla luodun arvioinnin perusteella voidaan tehdä eri johtopäätöksiä riippuen arvioinnin kohteesta ja arvioinnin tekevästä henkilöstä. MCDA-mallin avulla pyritään esittämään arvioinnin kohteeseen liittyvät sidokset mahdollisimman subjektiivisesti. Tämän työn tulosten arvioinnissa käytetään MCDA-mallin mukaista MAGIQ-menetelmää (engl. Multi-Attribute Global Inference of Quality). MAGIQ on analyyttinen hierarkiaprosessi -menetelmä. Analyyttiset hierarkiaprosessimenetelmät ovat yleisimmin käytettyjä MCDA-menetelmiä ja niiden avulla on mahdollista tehdä optimoituja ratkaisuja. MAGIQ-menetelmässä vaihtoehdot ja niiden ominaisuudet asetetaan taulukkoon, jossa niille annetaan numeroarvo. Ominaisuuksille on myös mahdollista antaa painoarvot jos niiden merkitystä halutaan korostaa tai pienentää. Tässä työssä käytetään MAGIQ-mallin yksinkertaistusta, jossa taulukkoon kirjataan ominaisuuksista vain tieto siitä, onko ominaisuus toteutettavissa kyseisen teknologian avulla. (Vrt. Tebest, 2010, s. 85; Vrt. Kiker et al., 2009, ss. 95-108; Vrt. McCaffrey, 2009, ss. 738-742; Vrt. Verta, 2006, ss. 39-41).

7.2 Ominaisuuksien vertailu

Taulukossa 7.1 on esitetty pilvipalvelintoteutuksen ja tavallisen, yhden palvelimen järjestelmän eroja avainominaisuuksien suhteen. Taulukossa pilvipalvelintoteutuksen vertailuarvoina käytetään Amazonin infrastruktuuri palveluna (IaaS) -tyyppistä ympäristöä.

Taulukko 7.1: Pilvipalvelintoteutuksen ja tavallisen palvelimen vertailutaulukko (Vrt. Brilliant Thinking, 2009)

	Pilvipalvelintoteutus	**Tavallinen palvelin**
Palvelinlaitteisto		
Laitteisto omassa hallussa	Ei	Kyllä / Ei
Korkea vikasietoisuus	Kyllä	Ei
Automaattinen skaalautuvuus	Kyllä	Ei
Vastuu laitteistosta ja ylläpidosta	Ei	Kyllä / Ei
Turvallisuus ja lakiteknisyys		
Tietoturvallinen ulkoisen yksityisen datan säilytys	Ei	Kyllä / Ei
Yhteensopivuus organisaation hallintotavan kanssa	Kyllä / Ei	Kyllä
Hinnoittelu		
Kiinteä kuukausihinta	Ei	Kyllä / Ei
Kustannustehokkuus	Kyllä	Kyllä / Ei

7.3 Skaalautuvuus

Yksi tärkeimmistä pilvipalvelinympäristön tarjoamista ominaisuuksista on palvelinalustan skaalautuvuus. Verkkopalvelun kävijämäärät saattavat päivätasolla olla hyvinkin pieniä, jolloin pärjätään pienelläkin laskentakapasiteetilla. Kuukauden aikana voi kuitenkin olla päiviä, jolloin verkkopalvelussa on iso piikki kävijämäärässä. Tämä voi johtua esimerkiksi jostain uudesta julkaisusta sivustolla tai vaikka mainoksesta jossakin mediassa. Näissä tilanteissa on tärkeää, että verkkopalvelu pystyy käsittelemään sivupyynnöt siedettävän aikarajan sisällä. Raja-arvoksi voidaan asettaa esimerkiksi kolme sekuntia. Tällaiset tilanteet voivat siis olla etukäteen tiedossa, mutta ne voivat tulla myös aivan yllättäen. Mikäli tulevasta kävijämääräpiikistä tiedetään etukäteen, voidaan pilvipalvelinalustan aktiivisten palvelinten määrän minimiä nostaa. Tällöin käytettävissä olevan laskentatehon määrä ei koskaan tipu kovin alhaiseksi. Yllättävissä kävijäpiikeissä sen sijaan järjestelmä joutuu todelliseen testiin. Ennen kuin käytettävissä olevien palvelininstanssien keskimääräinen rasitus on noussut yli hälytysrajan, niin palvelimet voivat olla jo täysin ylirasitettuja. Tämän lisäksi laskentatehon lisäämisessä pilveen on aina pieni viive, joka syntyy uuden palvelininstanssin käynnistämisestä ja lisäämisestä kuormantasaajaan.

On olemassa tilanteita, joissa sovellusta ei ole mahdollista tai järkevää hajauttaa usealle eri palvelimelle. Tämä voi olla tekninen rajoite tai se voi ai-

heuttaa liian suuria ylläpidollisia haasteita. Esimerkiksi Drupal-sivuston master-tietokannan hajauttaminen usealle master-tietokantapalvelimelle on teoriassa mahdollista, mutta sen toteuttaminen ja ylläpitäminen on haastavaa. Drupal ei tätä oletuksena tue, joten useamman master-palvelimen toteutus pitäisi itse rakentaa. Tällöin sivusto selviää yhden master-tietokantapalvelimen rikkoutumisesta, mutta on olemassa vaara, että kokonaisia tietokantatransaktioita katoaa tai tietokannan binääriloki korruptoituu. Drupal-sivuston tapauksessa on siis ainakin ylläpidon ja vikasietoisuuden kannalta turvallisempi ratkaisu käyttää yhtä master-tietokantaa. Tällöin luonnollisesti herää kysymys siitä, että voiko yhden master-tietokannan järjestelmä sitten skaalautua kovin pitkälle? Drupal-sivuston tapauksessa suurin osa ja optimitilanteessa jokainen anonyymin käyttäjän tuottama tietokantakysely on vain luku -kysely. Tällaiset vain luku -kyselyt voidaan ohjata mille tahansa replikoidulle tietokantapalvelimelle, eivätkä ne rasita master-tietokantapalvelinta ollenkaan. Master-tietokantapalvelimen on kuitenkin syytä olla huomattavasti slave-tietokantapalvelimia tehokkaampi.

7.4 Vikasietoisuus

Pilvipalvelintoteutuksella on mahdollista saavuttaa huomattavasti tavallista palvelinta parempi vikasietoisuus. Pilvipalvelintoteutuksessa käytettävissä olevat palvelininstanssit rekisteröidään kuormantasaajalle. Kuormantasaaja ohjaa kyselyt tasaisesti kaikille käytettävissä oleville palvelininstansseille. Kuormantasaaja suorittaa taustalla koko ajan tarkistuksia, että onko palvelin vielä toiminnassa. Jos kuormantasaaja havaitsee palvelininstanssissa ongelman, estää se liikenteen ohjaamisen tälle palvelimelle.

Amazonin pilvipalvelinkeskukset on jaettu erillisiin saatavuusalueisiin. Palvelininstansseja on mahdollista käynnistää eri saatavuusalueihin, jolloin yksittäisellä saatavuusalueella sattunut mittava laitteisto-ongelma ei kuitenkaan aiheuta palvelulle katkosta. Myös Amazon RDS tarjoaa mahdollisuuden käynnistää tietokantapalvelin Multi-AZ-tilassa (engl. Multi Availability Zone). Tällöin tietokannan sisältö replikoidaan automaattisesti toisessa saatavuusalueessa olevalle palvelimelle. Jos ensisijaisella palvelimella havaitaan ongelma, ohjataan kaikki tietokantakyselyt tällöin varapalvelimelle. Optimitilanteessa verkkopalvelun toiminnalle ja loppukäyttäjälle tämä muutos ei näy mitenkään.

Laitteisto-ongelmien on lisäksi varauduttava erilaisiin tahallisiin haittoihin, kuten palvelunestohyökkäyksiin (DDoS, engl. Distributed Denial-of-Service). Amazon käyttää sovelluskohtaisia palvelunestohyökkäyksen esto- tai lievennystoimenpiteitä. Lisäksi palvelunestohyökkäysten vaikutusta pyritään lieventämään sillä, että Amazonin käyttämät verkot on hajautettu useille eri palveluntarjoajille monipuolisuuden

saavuttamiseksi. (Amazon Web Services, 2011, s. 10). Edes pilvipalvelinympäristöissä palvelunestohyökkäyksiä ei kuitenkaan pystytä kokonaan estämään. Uskomus siitä, että esimerkiksi Amazon EC2 -palvelimet olisivat erityisen vastustuskykyisiä palvelunestohyökkäyksiä vastaan, perustuu siihen, että pilvessä on käytössä erityisen suuri määrä siirtokaistaa. Monissa tapauksissa, vaikka siirtokaistaa riittäisi, liikenne kyseiselle palvelimelle estetään kuitenkin lopulta kokonaan. Tällöin on mahdollista, ettei palvelu ole enää ollenkaan käytössä, koska sillä ei ole palvelimia, joihin vielä saa yhteyden. Mikäli palvelua yritetään pitää palvelunestohyökkäyksestä huolimatta päällä, voi siitä seurata erittäin suuria kustannuksia. Automaattisen skaalautuvuuden kautta käytössä olevien palvelininstanssien määrä on maksimissa ja verkkoliikenteen määrä on erittäin suuri. Palvelunestohyökkäyksen tapauksessa onkin siis tehtävä kompromisseja kustannusten ja saavutettavuuden välillä. (Govshteyn, M., 2010).

7.5 Kustannustehokkuus

Tavallisella yksittäisellä vuokrattavalla palvelimella ja pilvipalvelintoteutuksella on hyvin suuri ero hinnoittelupolitiikan suhteen. Ensiksi mainitulla kuukauden kokonaishinta on usein tarkalleen tiedossa. Kuukausihintaan sisältyy itse palvelimen resurssien lisäksi usein myös rajaton verkkoliikenne. Pilvipalvelinjärjestelmän lopullista kuukausihintaa sen sijaan ei pysty varmasti sanomaan. Siitä voidaan esittää vain arvioita. Lopullinen hinta muodostuu useasta eri osasta. Siihen vaikuttavat muun muassa palvelininstanssien käytössä olevat resurssit, palvelininstanssien käyttötunnit, verkkoliikenteen määrä sekä levytilan käyttö.

Pilvipalvelintoteutuksen kustannustehokkuus muodostuu kuitenkin viime kädessä siitä, että käytössä olevien resurssien määrää optimoidaan muuttuvien tarpeiden mukaisesti. Jos tiedetään, että palvelinympäristöä tullaan käyttämään pitkään ja siinä tarvitaan aina tietyn verran resursseja, niin on kustannustehokasta varata nämä resurssit etukäteen usean vuoden sopimuksella. Tällöin instanssien tuntihinta on huomattavasti halvempi.

Optimoinnin kautta pilvipalvelintoteutuksesta maksettava kuukausihinta voi siis olla tavallista kiinteää palvelinta pienempi. Sen keskimääräinen käytössä oleva laskentakapasiteetti on huomattavasti pienempi kuin kiinteällä palvelimella, mutta se on mahdollista väliaikaisesti skaalata monin kerroin yksittäistä palvelinta tehokkaammaksi. Samalla saavutetaan myös muita etuja, kuten parempi vikasietoisuus ja parempi saavutettavuus.

7.6 Drupalin soveltuvuus pilvipalvelinympäristöön

Drupal-järjestelmän sovittaminen pilvipalvelinympäristöön ei ole aivan yksinkertaista. Tavallisessa yhden palvelimen toteutuksessa päivitysten asentamiseen tai Drupalin sivustokohtaisten tiedostojen hallintaan ei tarvitse kiinnittää erityistä huomiota. Pilvipalvelintoteutuksessa sen sijaan käytetään yhtä levykuvaa, josta monistetaan tarpeen mukaan uusia palvelininstansseja pilveen ja liitetään ne kuormantasaajaan. Jokaisen Drupal-instanssin pitää kuitenkin olla aina ajan tasalla, jotta sivusto toimisi oikein. Tiedostojen jakaminen instanssien välillä S3-säiliötä käyttäen aiheuttaa aina kuitenkin pientä ylimääräistä viivettä sekä verkkoliikennettä. Kaikkia Drupalin tiedostoja ei siis ole järkevää säilyttää yhteisessä S3-säiliössä, vaan ne on tehokkainta pitää instanssin omalla levyllä. Nämä tekijät aiheuttavatkin pilvipalvelinympäristössä uusia ylläpidollisia haasteita.

Isompien päivitysten ja muutosten tekeminen vaatii siis aina uuden levykuvan luomisen. Lisäksi päivitysten asentaminen jokaiselle jo käytössä olevalle palvelimelle levynkuvan päivittämisen lisäksi vaatii mahdollisen huoltoikkunan tai tarpeeseen soveltuvan työkalun. Tähän tarkoitukseen löytyy kuitenkin useitakin työkaluja. Esimerkiksi puppet[26]-työkalun avulla on mahdollista suorittaa automatisoituja toimenpiteitä jokaisella instanssilla.

Drupalin asentaminen pilvipalvelinympäristöön on siis mahdollista, mutta järjestelmän kompleksisuus kasvaa äkkiä epämiellyttävän suureksi. Täydellisen automaation toteuttaminen vaatii huomattavan paljon skriptausta. Päivitysten asentaminen ilman erillistä huoltoikkunaa aiheuttaa potentiaalisia vaaratilanteita, kun päivitettävät tiedostot eivät ole ehtineet samanaikaisesti synkronoitua jokaiselle palvelimelle. Yksittäinen instanssi voi jäädä suorittamaan jotain loputonta silmukkaa, jolloin se hidastuu huomattavasti. Kuormantasaaja ei kuitenkaan välttämättä luokittele tätä instanssia vielä rikkinäiseksi. Tällaisessa tapauksessa tästä yksittäisestä instanssista voi muodostua selkeä pullonkaula, joka hidastaa jokaista sivulatausta. Koko järjestelmän kannalta yhdeksi sen heikkoudeksi nousee siis käytettyjen eri komponenttien suuri lukumäärä. Jokaisen komponentin, eli muun muassa palvelimien, tietokantapalvelimien, keskitetyn levyjärjestelmän sekä sisällönjakeluverkon on toimittava täysin, että sivusto on käytettävissä. Yksittäisen komponentin vikaantuminen voi pahimmassa tapauksessa kaataa koko järjestelmän.

[26] `http://puppetlabs.com/puppet/what-is-puppet/`

8 Yhteenveto

Pilvipalvelut ja pilvilaskenta ovat ajankohtainen tutkimusalue. Pilvipalveluiden käyttäjien määrä on kasvanut räjähdysmäisesti viime vuosina. Tässä työssä pilvipalveluiden teoriaa ja sovelluksia esitellään kattavasti. Työssä havaittiin, että pilvipalveluiden avulla on mahdollista toteuttaa kustannustehokkaasti skaalautuva ja erittäin vikasietoinen palvelinalusta. Työssä havaittiin lisäksi, että verkkopalveluiden tapauksessa skaalautuvuutta voidaan tukea myös käyttämällä erillistä sisällönjakeluverkkoa. Sisällönjakeluverkon avulla kaikki verkkopalvelun staattinen tieto voidaan jakaa loppukäyttäjille strategisesti ympäri maailmaa sijoitetuilta sisällönjakeluverkon reunapalvelimilta. Koko järjestelmän skaalautuvuuden parantumisen lisäksi myös loppukäyttäjän näkökulmasta palvelun laatu on parempi, kun tiedostojen latausajat ovat pienemmät.

Pilvipalveluita tarjoavia palveluntarjoajia on useita. Tässä työssä ei pyritty vertailemaan useita pilvipalveluntarjoajia keskenään, vaan valittiin yksi palveluntarjoaja tarkempaan tarkasteluun. Amazon Web Services oli helppo valinta käyttäjämäärien perusteella, sillä se on kaikkein suurin. Työssä esiteltiin kattavasti koko Amazonin pilvipalvelutarjontaa. Amazonin hinnoittelumalli poikkeaa huomattavasti tavallisesta virtuaalipalvelimen hinnoittelusta. Jokainen komponentti on hinnoiteltu erikseen käytön mukaan. Työssä pohdittiin esimerkkikokoonpanon avulla kuukausittaista kustannusarviota.

Diplomityössä toteutettiin Amazonin pilvipalvelinjärjestelmän avulla kustannustehokkaasti skaalautuva ja vikasietoinen palvelinympäristö. Pilvipalvelinympäristön pohjaksi luotiin yksi palvelininstanssi, johon asennettiin käyttöjärjestelmä sekä kaikki tarvittavat sovellukset. Tämä vaihe ei vielä eronnut mitenkään tavallisesta virtuaalipalvelimen asennuksesta. Ero tavalliseen virtuaalipalvelimeen syntyy kuitenkin nopeasti, kun otetaan käyttöön järjestelmän automaattinen skaalautuvuus. Automaattinen skaalautuvuus tarkoittaa sitä, että järjestelmän rasitusta seurataan erilaisilla mittausjärjestelmillä ja mittausarvojen perusteella käytössä olevaa laskentakapasiteettia skaalataan joko ylös- tai alaspäin. Kustannustehokkuus syntyy siitä, että käytössä on aina vain juuri sen verran laskentatehoa kuin mitä järjestelmä sillä hetkellä vaatii. Työssä esiteltiin kattavasti, miten järjestelmän automaattinen skaalautuvuus voidaan Amazonin pilvipalvelinympäristössä ottaa käyttöön. Alkuperäisellä palvelininstanssilla on keskeinen rooli skaalautuvuuden toteuttamisessa. Siitä luodaan levykuva, joka on siis täydellinen kopio palvelimen sisällöstä. Levykuvan avulla voidaan tarvittaessa käynnistää uusi palvelininstanssi, joka liitetään kuormantasaajaan. Kuormantasaajan avulla sivupyynnöt ohjataan tasaisesti kaikille käytössä oleville palvelininstansseille. Näin uusien palvelininstanssien avulla voidaan pilven laskentakapasiteettia kasvattaa lähes rajatta. Kun

ylimääräiselle palvelininstanssille ei ole enää tarvetta, niin palvelin sammutetaan ja kaikki sen sisältö poistetaan.

Pilvipalvelinympäristön käytöstä syntyy myös useita erilaista haasteita. Yksi keskeisin haaste liittyy tiedostojen tallettamiseen palvelimen paikalliselle levylle. Automaattisen skaalautuvuuden myötä uusia palvelininstansseja luodaan ja poistetaan automaattisesti. On siis erittäin tärkeää varmistaa, ettei palvelimen paikalliselle levylle tallennettuja tiedostoja tuhota lopullisesti, kun palvelin sammutetaan. Lisäksi palvelimille tallennettujen tiedostojen tulee olla samanaikaisesti jokaisen palvelininstanssin käytössä. Työssä esitellään tapa, jolla tiedostot on mahdollista tallettaa keskitettyyn ja varmuuskopioituun sijaintiin, josta jokainen palvelininstanssi voi ne noutaa. Lisäksi samaa tietovarastoa voidaan käyttää sisällönjakeluverkon toteutuksessa.

Lisäksi työssä esitettiin, miten Drupal-sisällönhallintajärjestelmä on mahdollista asentaa luotuun pilvipalvelinympäristöön. Tavoitteena oli saada Drupal käyttämään hyödykseen kaikkia pilvipalvelinympäristön tarjoamia resursseja mahdollisimman tehokkaasti. Aikaisemmin esitetyt haasteet liittyvät myös vahvasti Drupalin toimintaan. Drupal tallettaa paljon tiedostoja palvelimen paikalliselle levylle. Lisäksi kaikkien käytössä olevien palvelininstanssien toiminnan tulee olla identtistä. Järjestelmän toiminnan kannalta on siis kriittisen tärkeää, että palvelininstanssien tiedostot on keskitetty yhteen sijaintiin. Drupal ei osaa itse oletuksena hyödyntää sisällönjakeluverkkoa, vaan sen hyödyntämiseksi on asennettava kolmannen osapuolen moduuli. Moduulin avulla on mahdollista erottaa staattiset ja dynaamiset tiedostot toisistaan, ja jakaa staattiset tiedostot käyttäen sisällönjakeluverkkoa.

Lähteet

Amazon ELB. (2012). *Elastic Load Balancing.* Lainattu 1. elokuuta, 2012, saatavilla `http://aws.amazon.com/elasticloadbalancing/`

Amazon Web Services. (2011). *Amazon Web Services: Overview of Security Process.* Lainattu 13. joulukuuta, 2012, saatavilla `http://s3.amazonaws.com/aws_blog/AWS_Security_Whitepaper_2008_09.pdf`

Amazon Web Services Blog. (2009). *AWS Management Console - Now with Amazon CloudWatch Support.* Lainattu 9. joulukuuta, 2012, saatavilla `http://aws.typepad.com/aws/2009/08/aws-management-console-now-with-amazon-cloudwatch-support.html`

Barrett, D., & Kipper, G. (2010). *Virtualization and Forensics - A Digital Forensic Investigator's Guide to Virtual Environments.* Elsevier, Inc. (ISBN: 978-1-59749-557-8).

Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., & Sturm, W. (2002). CDN brokering. *Computer Communications*, *25*(4), 393–402.

Brilliant Thinking. (2009). *Cloud vs. Traditional Hosting.* Lainattu 13. joulukuuta, 2012, saatavilla `http://www.brilliantthinking.net/2009/10/16/cloud-vs-traditional-hosting/`

Buyya, R., Pathan, M., & Vakali, A. (2008). *Content Delivery Networks.* Springer. (ISBN: 978-3-540-77886-8).

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., & Sitaraman, B., R. Weihl. (2002). Globally Distibuted Content Delivery. *Internet Computing, IEEE*, *6*, 50–58.

Dillon, T., Chen, W., & Chang, E. (2010). Cloud Computing: Issues and Challenges. *Advanced Information Networking And Applications (AINA), 2010 24th IEEE International Conference*, 27–33.

Douglas, H., & Gehrmann, C. (2009). *Secure Virtualization and Multicore Platforms State-of-the-Art report.* Swedish Institute of Computer Science.

Dubey, A., & Wagle, D. (2007). *Delivering software as a service.* The McKinsley Quartely.

Ecommerce Developer. (2011). *Use Content Delivery Networks for Better Performance.* Lainattu 17. lokakuuta, 2012, saatavilla `http://ecommercedeveloper.com/articles/2784-use-content-delivery-networks-for-better-performance/`

Garrett, J. (2005). *Ajax: A New Approach to Web Applications.* Lainattu 9. joulukuuta, 2012, saatavilla `http://www.robertspahr.com/teaching/nmp/ajax_web_applications.pdf`

Garson, D. (2008). *Case studies.* North Carolina State University. Lainattu 12. elokuuta, 2012, saatavilla `http://www.statisticalassociates.com/casestudies.html`

Goldberg, R. (1973). *Architectural Principles for Virtual Computer Systems.* Harvard University, Electronic Systems Division.

Google. (2012). *Google App Engine: Why App Engine.* Lainattu 12. elokuuta, 2012, saatavilla `https://developers.gogole.com/appengine/whyappengine`

Govshteyn, M. (2010). *Secure Cloud Review: Cloud protection from DDoS attacks only slighly more effective than snake oil.* Lainattu 13. joulukuuta, 2012, saatavilla `http://securecloudreview.com/2010/12/cloud-protection-from-ddos-attacks-only-slightly-more-effective-than-snake-oil/`

Grossman, R. (2009). The Case for Cloud Computing. , *11* (2), 23–27.

Henderson, C. (2006). *Building Scalable Web Sites.* O'Reilly (ISBN: 978-0-596-10235-7).

How CRM Works. (2010). *Cloud Computing - Deployment Models.* Lainattu 1. marraskuuta, 2012, saatavilla `http://howcrmworks.com/2010/08/cloud-computing-deployment-models/`

IaaS Definition. (2012). *IaaS Definition.* Lainattu 12. elokuuta, 2012, saatavilla `http://www.iaasdefinition.com/`

IBM. (2012). *Pilvipalvelut.* Lainattu 12. elokuuta, 2012, saatavilla `http://www-05.ibm.com/fi/solutions/cloud/`

Järvinen, P., & Järvinen, A. (2004). *Tutkimustyön metodeista.* Opinpajan Kirja, Tampere. (ISBN: 952-99233-2-5).

Kiker, G., Bridges, T. S., Varghese, A., Seager, T. P., & Linkov, I. (2009). Application of multicriteria decision analysis in environmental decision making. *Integrated Environmental Assessment and Management*, *1*(2), 95–108.

Krishnamurthy, B., Wills, C., & Zhang, Y. (2001). On the Use and Performance of Content Distribution Networks. *Proceeding IMW '01 Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 169–182.

Linthicum, D. (2009). *Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide.* Addison-Wesley Professional. (ISBN: 978-0136009221).

Liu, C., & Albitz, P. (2006). *DNS and BIND, 5th Edition.* O'Reilly Media. (ISBN: 978-0-596-10057-5).

Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud Security and Privacy.* O'Reilly Media, Inc. (ISBN: 978-0-596-80276-9).

McCaffrey, J. D. (2009). Using the Multi-Attribute Global Inference of Quality (MAGIQ) Technique for Software Testing. *2009 Sixth International Conference on Information Technology: New Generations. IEEE Computer Society. (ISBN: 978-0-7695-3596-8).*

Mockapetris, P. (1987). *Domain Names - Implementation and Specification.* Lainattu 19. tammikuuta, 2013, saatavilla `http://tools.ietf.org/html/rfc1035`

Mulerikkal, J. P., & Khalil, I. (2007). An Architecture for Distributed Content Delivery Network. *Networks, 2007. ICON 2007. 15th IEEE International Conference*, 359–364.

MySQL Replication. (2012). *MySQL 5.0 Reference Manual, Chapter 16: Replication.* Lainattu 1. elokuuta, 2012, saatavilla `http://dev.mysql.com/doc/refman/5.0/en/replication.html`

Ricca, F., & Tonella, P. (2001). Analysis and testing of Web applications. *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, 25–34.

Salo, I. (2010). *Cloud computing - palvelut verkossa.* Docendo. (ISBN: 978-951-0-36584-7).

Scheffy, C. (2007). *Virtualization For Dummies, AMD Special Edition.* Wiley Publishing, Inc. (ISBN: 978-0-470-13156-5).

Tebest, T. (2010). *Verkkopalvelun käytön seuranta ja seurantatiedon visualisointi.* Diplomityö. Saatavilla http://matriisi.ee.tut.fi/hypermedia/julkaisut/di_teemo_tebest.pdf.

Vakali, A., & Pallis, G. (2003). Content Delivery Networks: Status and Trends. *Internet Computing, IEEE*, *7*(6), 68–74.

VanDyk, J., & Westgate, M. (2007). *Pro Drupal Development.* Apress.

Verta, O. (2006). *Päätösanalyysi vuorovaikutteisen suunnittelun tukena: Tapaustarkastelussa Koitereen säännöstelyn monitavoitteinen kehittäminen.* Diplomityö. Saatavilla http://water.tkk.fi/wr/tutkimus/thesis/Verta2006.pdf.

Webopedia. (2012). *Disk image.* Lainattu 10. joulukuuta, 2012, saatavilla http://www.webopedia.com/TERM/D/disk_image.html

Xu, Z., Hu, Y., & Bhuyan, L. (2006). Efficient Server Cooperation Mechanism in Content Delivery Network. *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*, 8–440.

Youseff, L., Butrico, M., & Da Silva, D. (2008, 12-16 Nov.). Toward a Unified Ontology of Cloud Computing. Teoksessa M. Pierce (toim.), *Grid Computing Environments Workshop, 2008. GCE '08* (s. 1–10). Univ. of California, Santa Barbara, Santa Barbara, CA, USA.